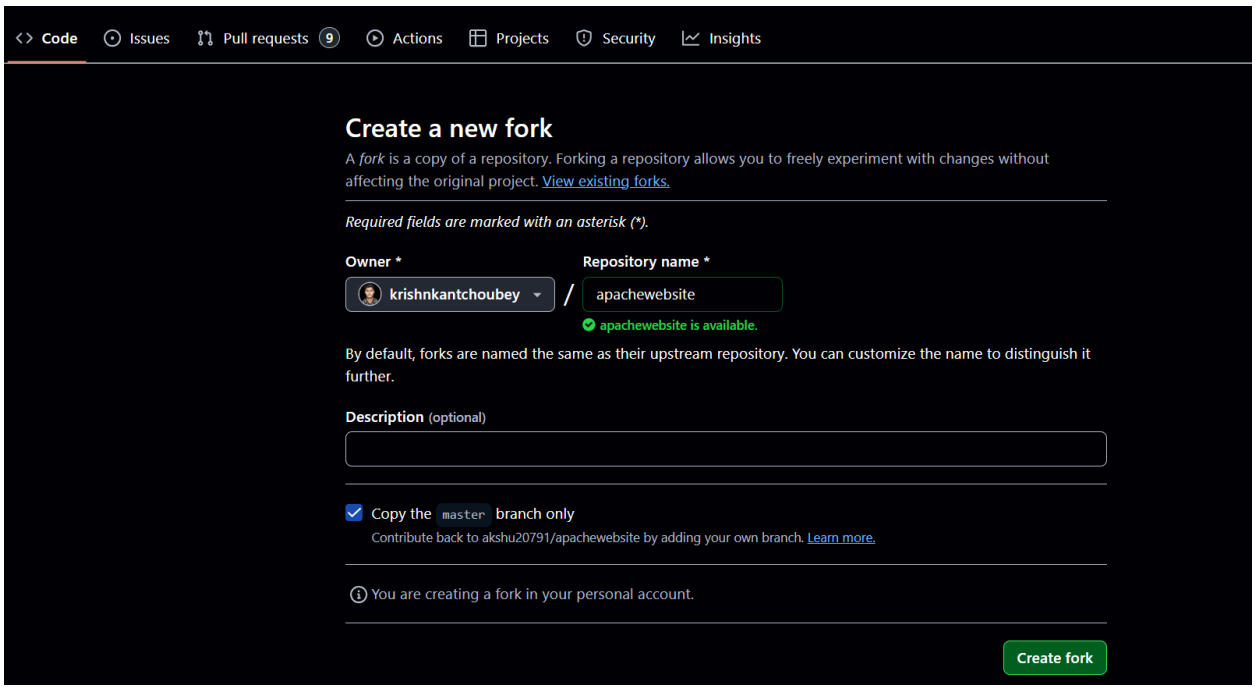
# Deploying Apache Project using Ansible via Jenkins

**Aim :** The project aims to Deploy Apache project on Node EC2's using Ansible which will be triggered via Jenkins pipeline

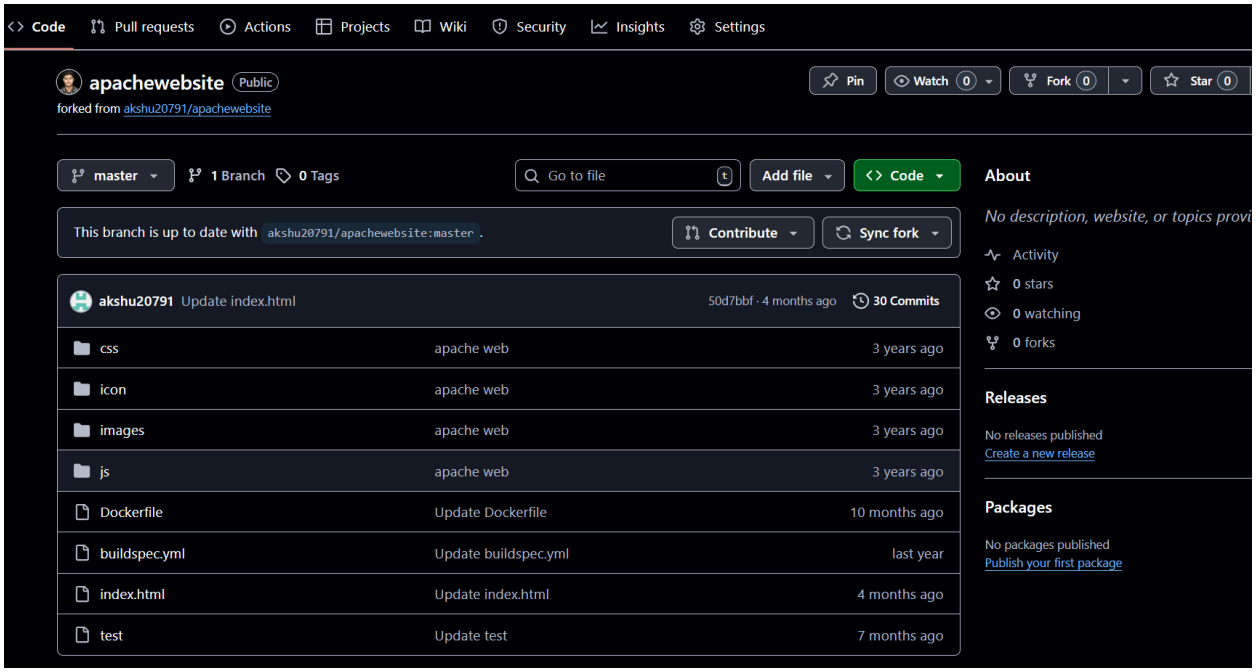
## Git Hub repo to fork :

https://github.com/akshu20791/apachewebsite




## Git Hub repo for reference :

krishnkantchoubey/apachewebsite




## Base plan :

1.Create a Master EC2 with – Jenkins, Ansible, docker installed

2. Create Cluster and nodes

3.Connect the Node to master

4.Make a Jenkins pipeline and deploy Apache project on all nodes by running Ansible playbook

## Steps to Follow:

### 1. Create a Master EC2 Instance

### 2. Create an IAM User

### 3. Install Required Packages and Tools

## 4. Create the EKS Cluster

## 5. Configure and Establish Jenkins

## 6. Install Necessary Plugins in Jenkins

## 7. Set Up Dynamic Inventory for Ansible

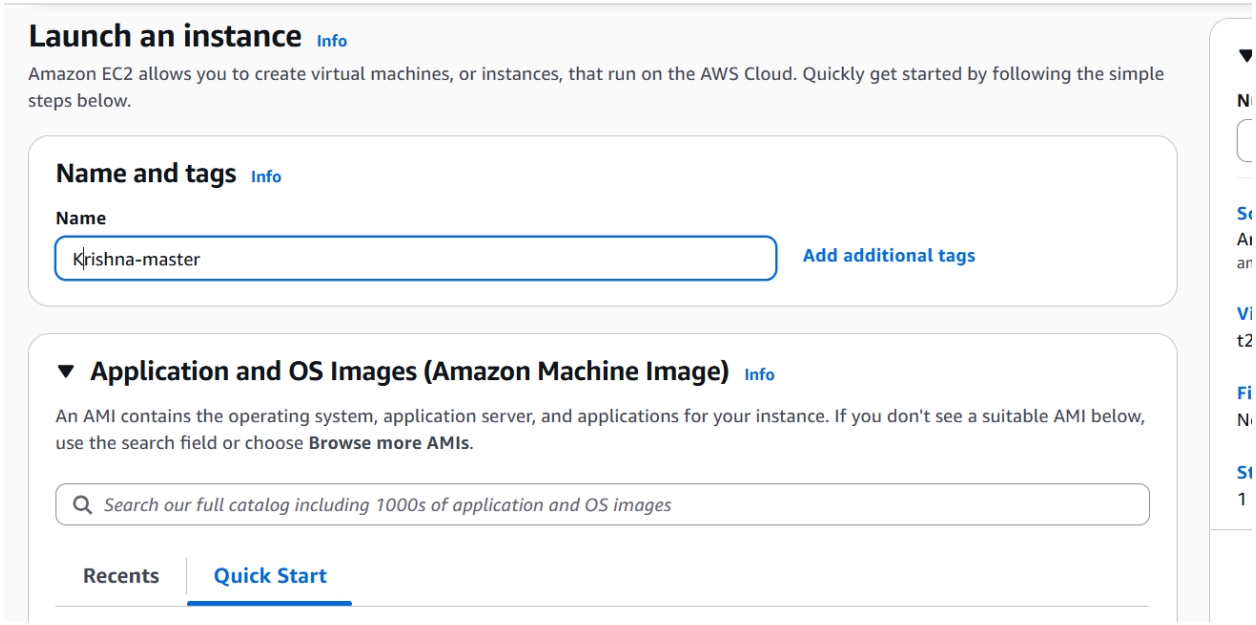
## 8. Connect Ansible Master with Nodes

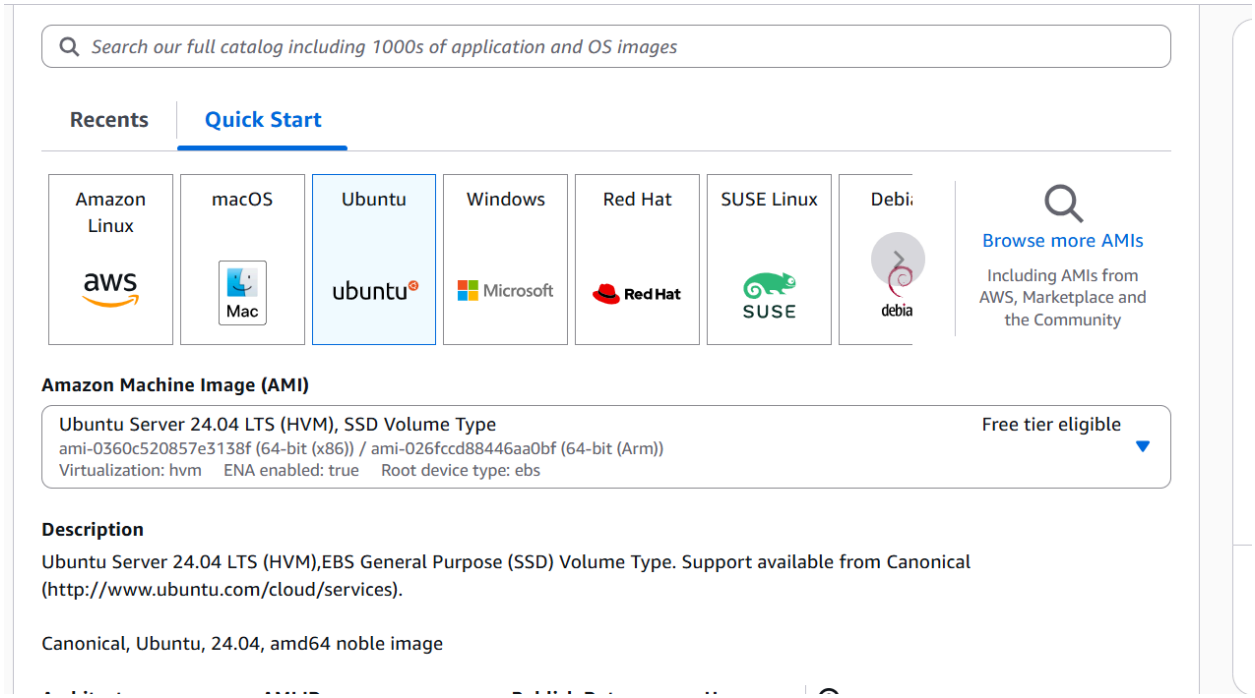
## 9. Execute the Final Deployment Using Jenkins Pipeline

## 10. Verify and Confirm Successful Deployment

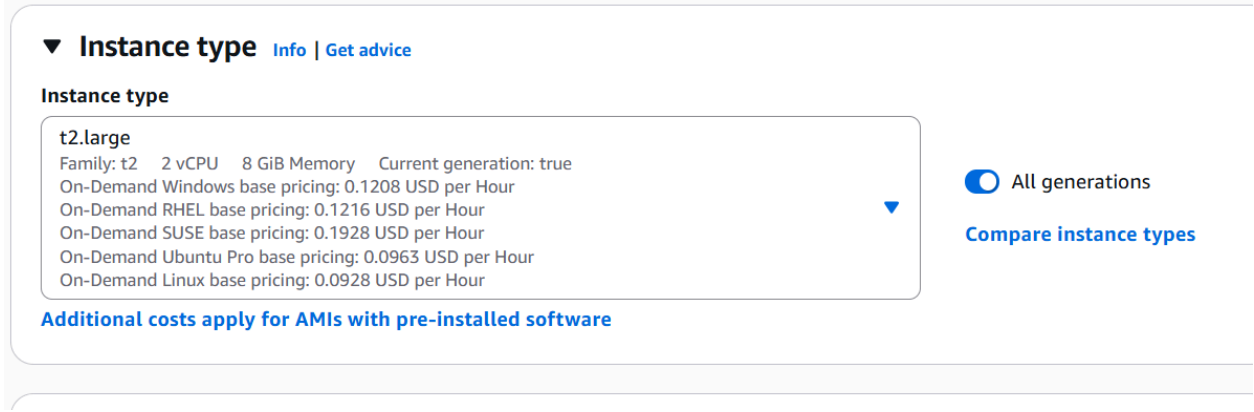
# Step 1. Create a Master EC2

a) Give name to EC2 and select AMI as Ubuntu 24.04LTS

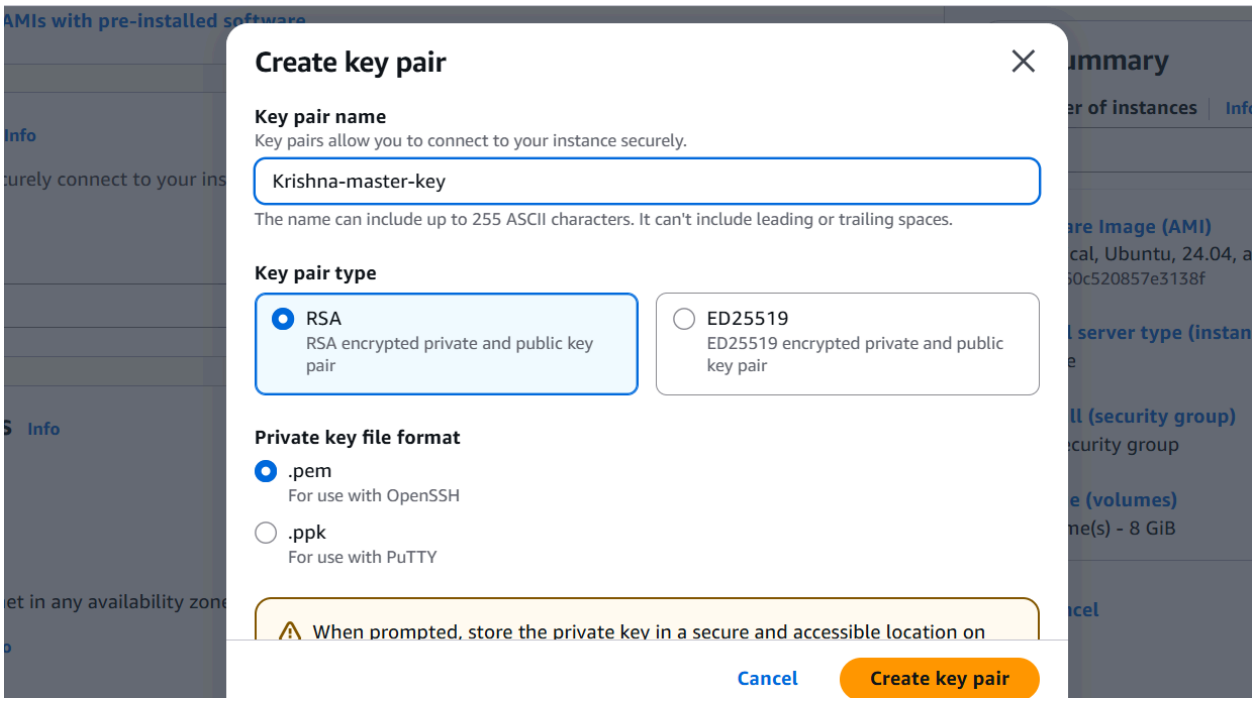





b) Select instance type as t2.large ( since this is master EC2 it will require large size )




c) Create a Key pair ( Save it we will use this key pair throughout the project )
Give it name -> key pair type = RSA -> Key file format = .pem -> Create key pair




## d) Create Security Group

1. Click on **Create Security Group**.

2. Enter the following details:

    - **Name**: Provide a suitable name for the security group.

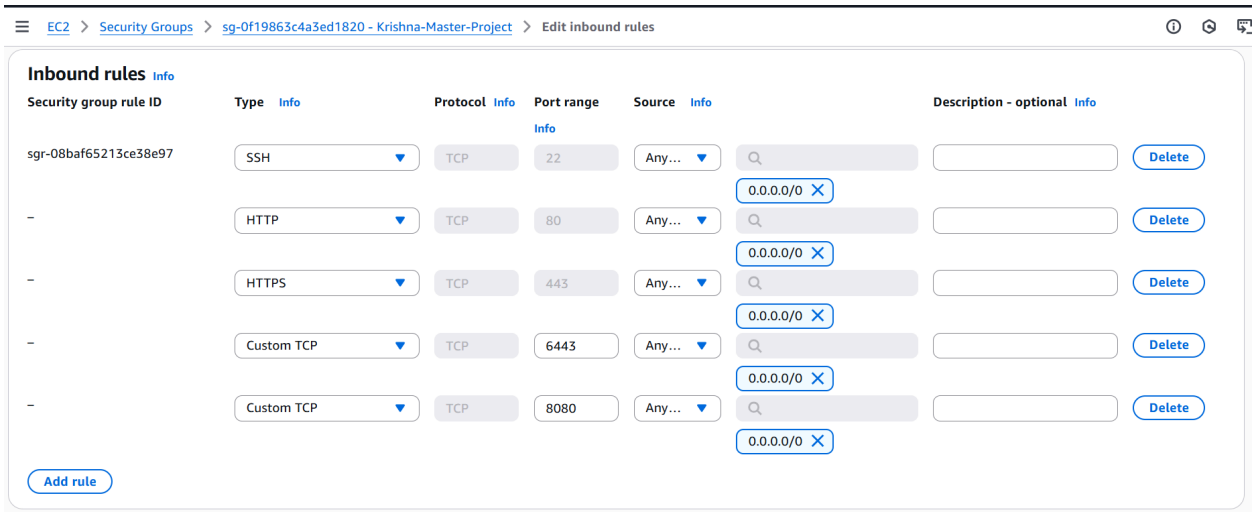
    - **Description**: Enter a brief description.

    - **VPC**: Select the **default VPC**.

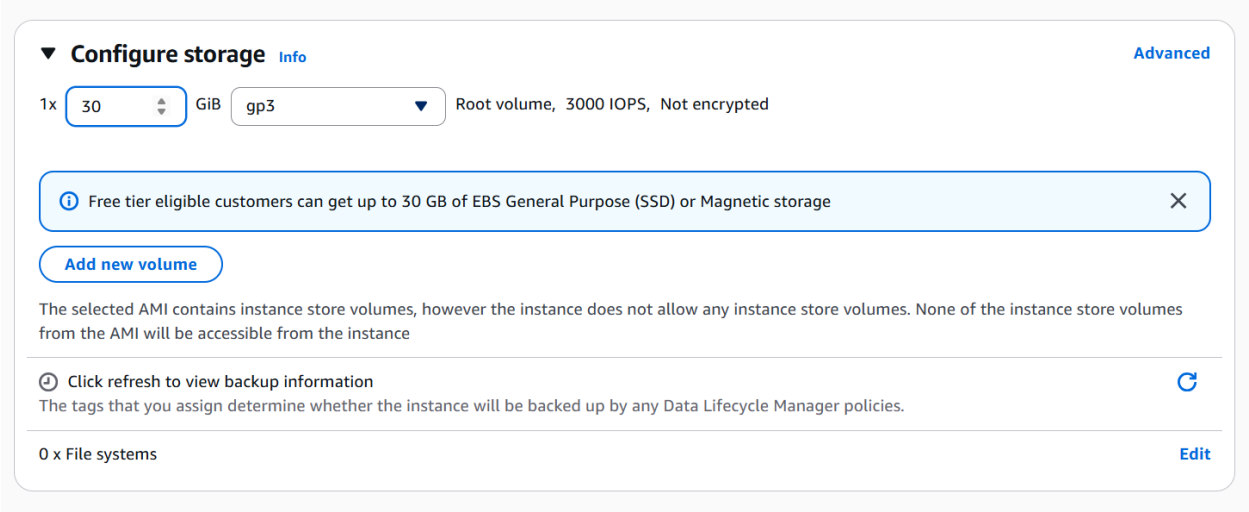
3. Add the following **Inbound rules**:

| Type | Protocol | Port Range | Source | Description |
|---|---|---|---|---|
| SSH | TCP | 22 | 0.0.0.0/0 | Allow SSH access |
| HTTP | TCP | 80 | 0.0.0.0/0 | Allow web traffic |
| HTTPS | TCP | 443 | 0.0.0.0/0 | Allow secure web traffic |
| Custom TCP Rule | TCP | 6443 | 0.0.0.0/0 | For Kubernetes API access |
| Custom TCP Rule | TCP | 8080 | 0.0.0.0/0 | For Jenkins access |

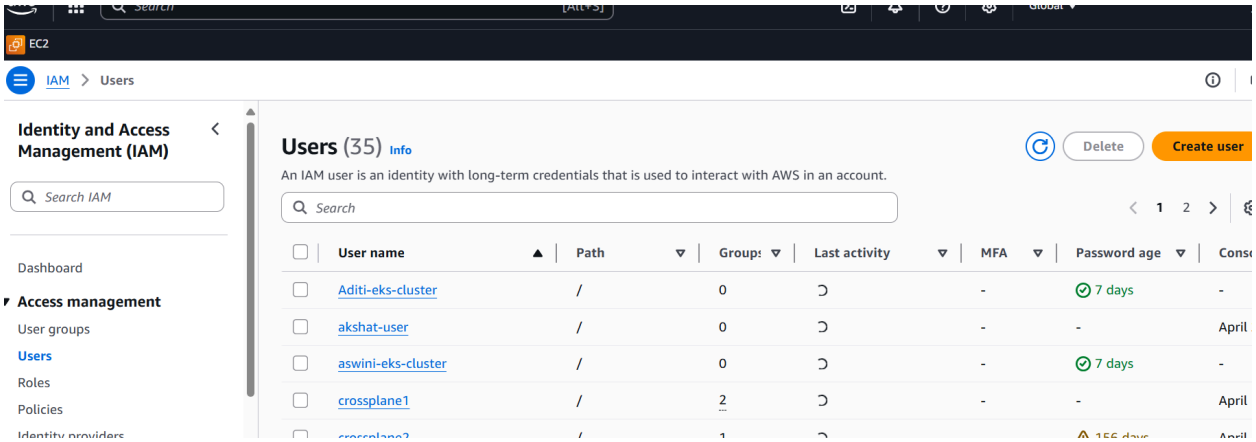
4.Click **Create Security Group** to finalize



e) Configure Storage as 1x30 GiB gp3 then click on Launch Instance

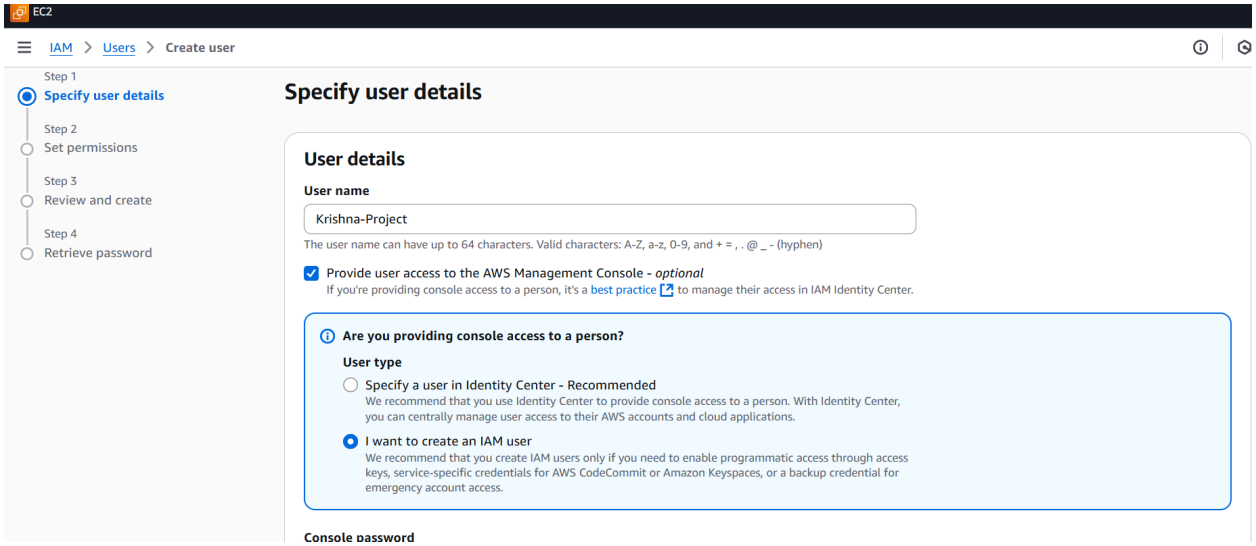


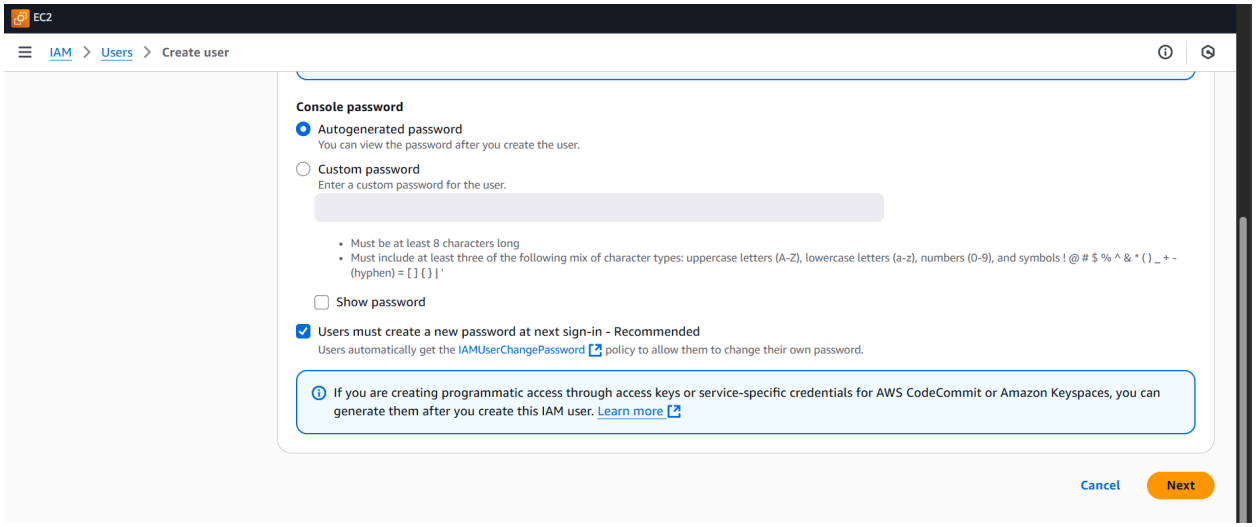

## Step 2 . Create IAM User



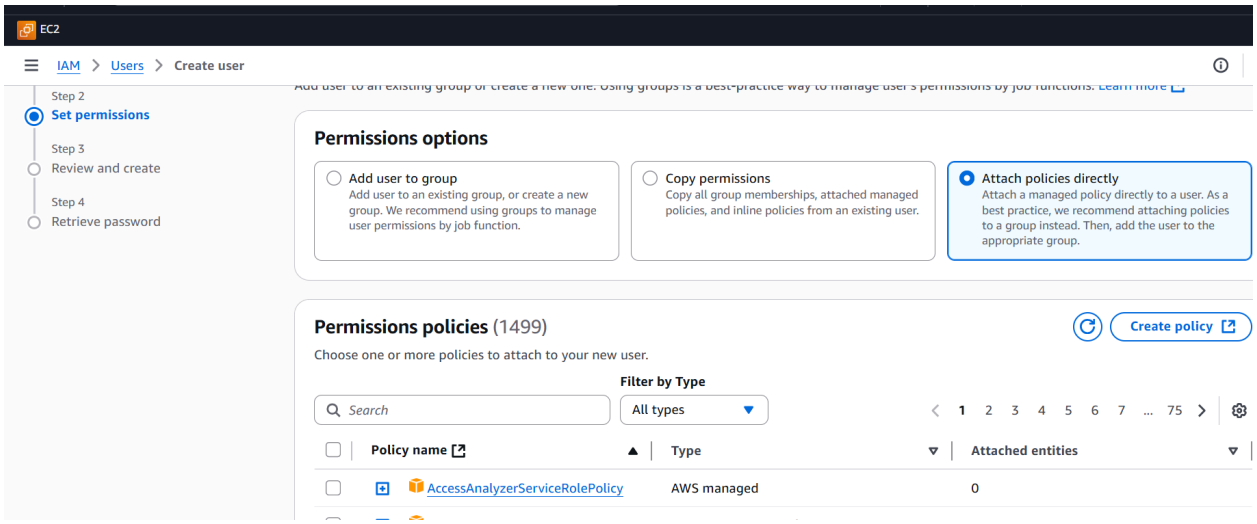

## b) Create IAM User

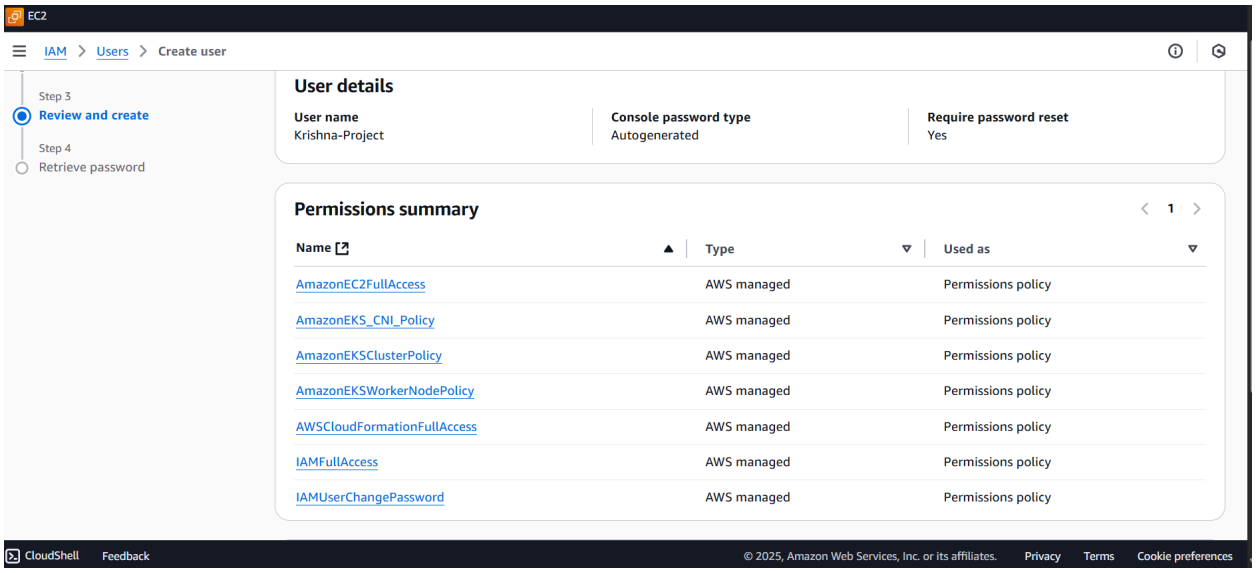
1. Enter the **Name** for the IAM user.
2. Check **Provide user access**.
3. Select **I want to create an IAM user**.
4. Click **Custom password** and enter the desired password.
5. Uncheck **User must create a new password at next sign-in**.
6. Click **Next**.
7. Skip all additional settings and click **Next** again.
8. Click **Create user** to complete the process.



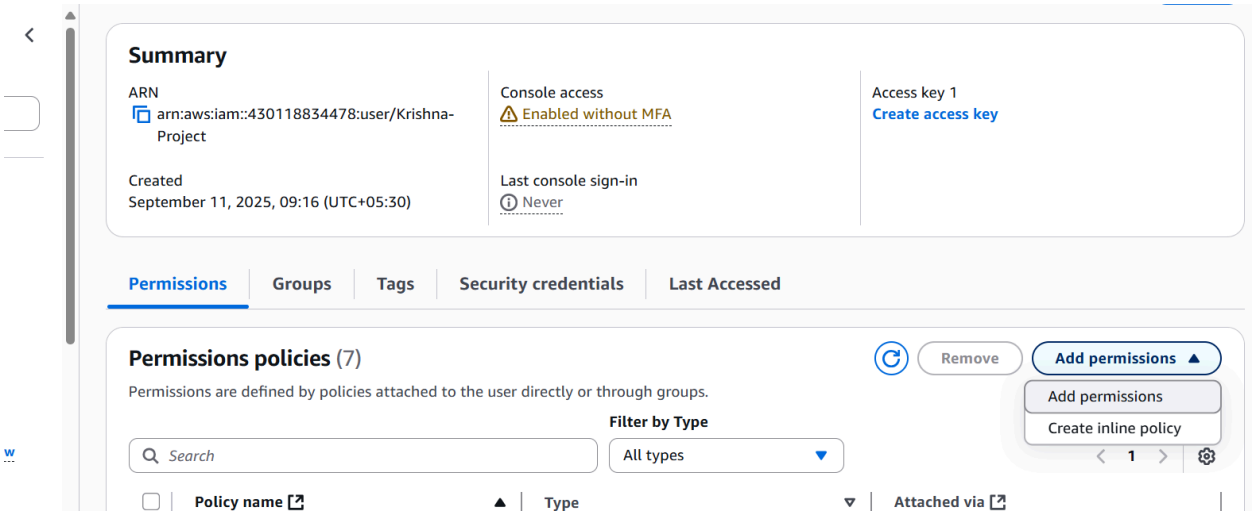



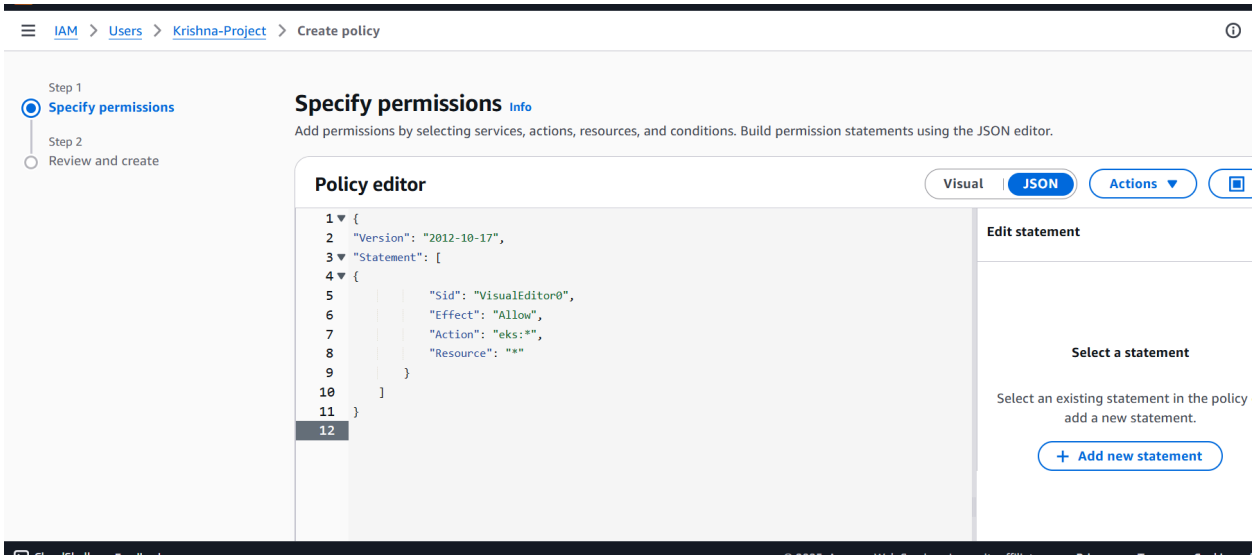

c) Move back to user list and select your user -> click Add Permissions -> Click on Attach Policies Directly

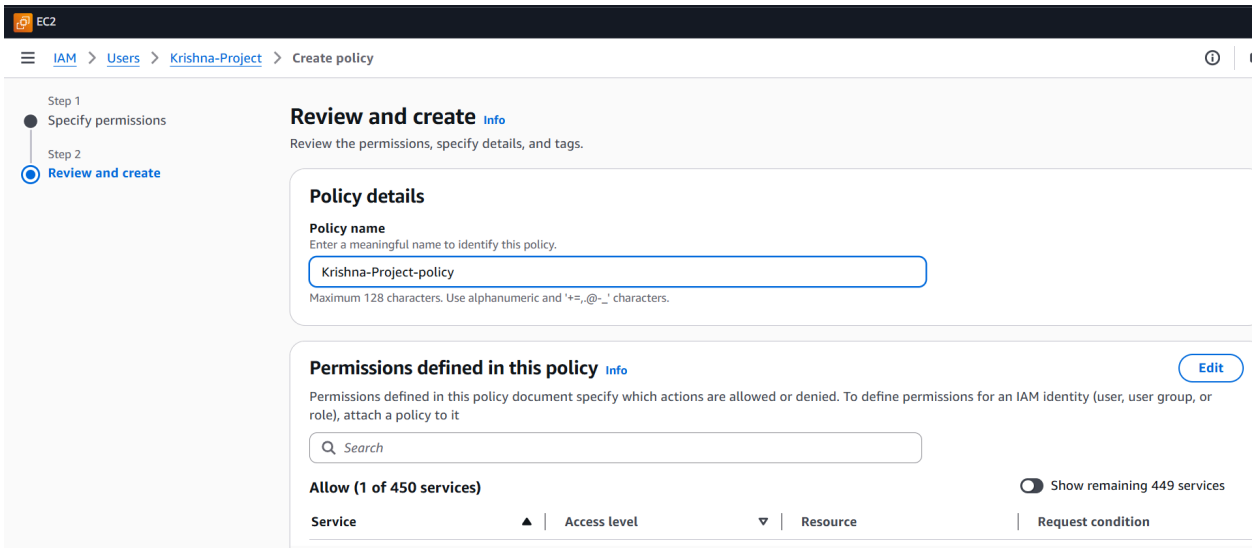


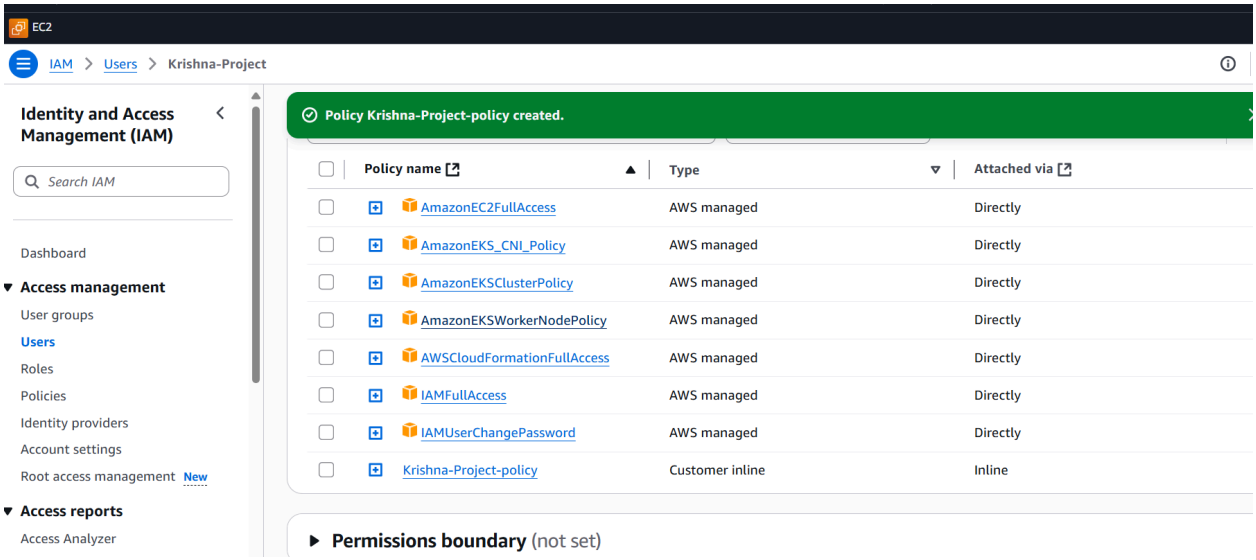

Click on Add Permission -> Create inline policy


Summary
ARN
arn:aws:iam::430118834478:user/Krishna-Project
Console access
Enabled without MFA
Access key 1
Create access key
Created
September 11, 2025, 09:16 (UTC+05:30)
Last console sign-in
Never
Permissions
Groups
Tags
Security credentials
Last Accessed
Permissions policies (7)
Permissions are defined by policies attached to the user directly or through groups.
Remove
Add permissions
Add permissions
Create inline policy
Filter by Type
Search
All types
Policy name
Type
Attached via


IAM > Users > Krishna-Project > Create policy
Step 1
Specify permissions
Step 2
Review and create
Specify permissions Info
Add permissions by selecting services, actions, resources, and conditions. Build permission statements using the JSON editor.
Policy editor
Visual
JSON
Actions
{
"Version": "2012-10-17",
"Statement": [
{
"Sid": "VisualEditor0",
"Effect": "Allow",
"Action": "eks:*",
"Resource": "*"
}
]
}
Edit statement
Select a statement
Select an existing statement in the policy or add a new statement.
Add new statement

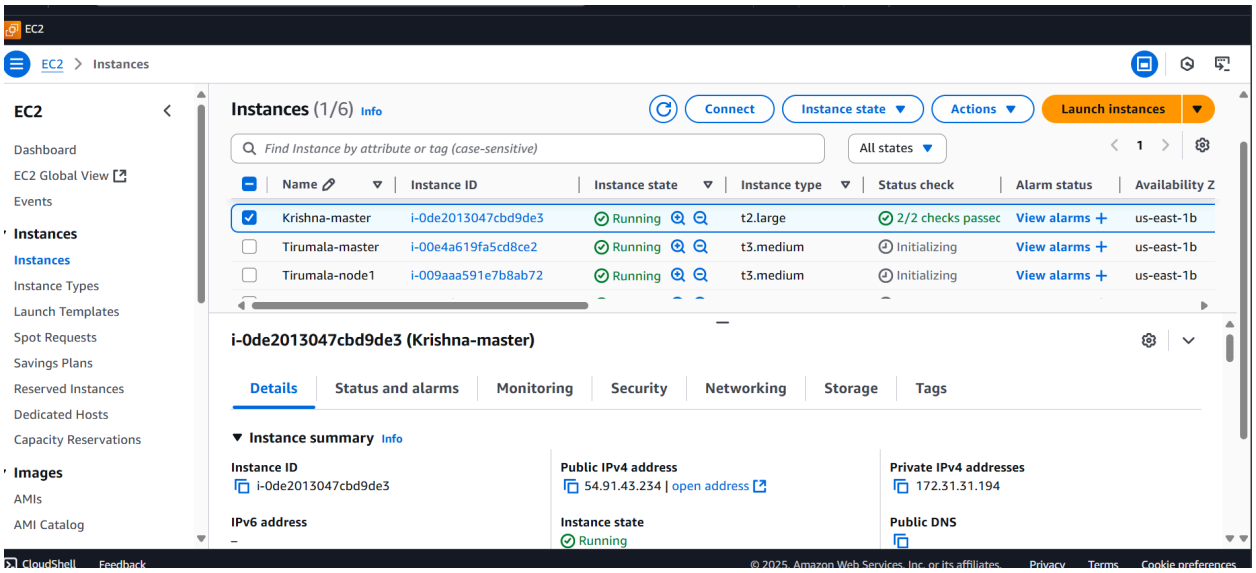

EC2
IAM > Users > Krishna-Project > Create policy
Step 1
Specify permissions
Step 2
Review and create
Review and create Info
Review the permissions, specify details, and tags.
Policy details
Policy name
Enter a meaningful name to identify this policy.
Krishna-Project-policy
Maximum 128 characters. Use alphanumeric and '+=,.@-_' characters.
Permissions defined in this policy Info
Edit
Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it
Search
Allow (1 of 450 services)
Show remaining 449 services
Service
Access level
Resource
Request condition

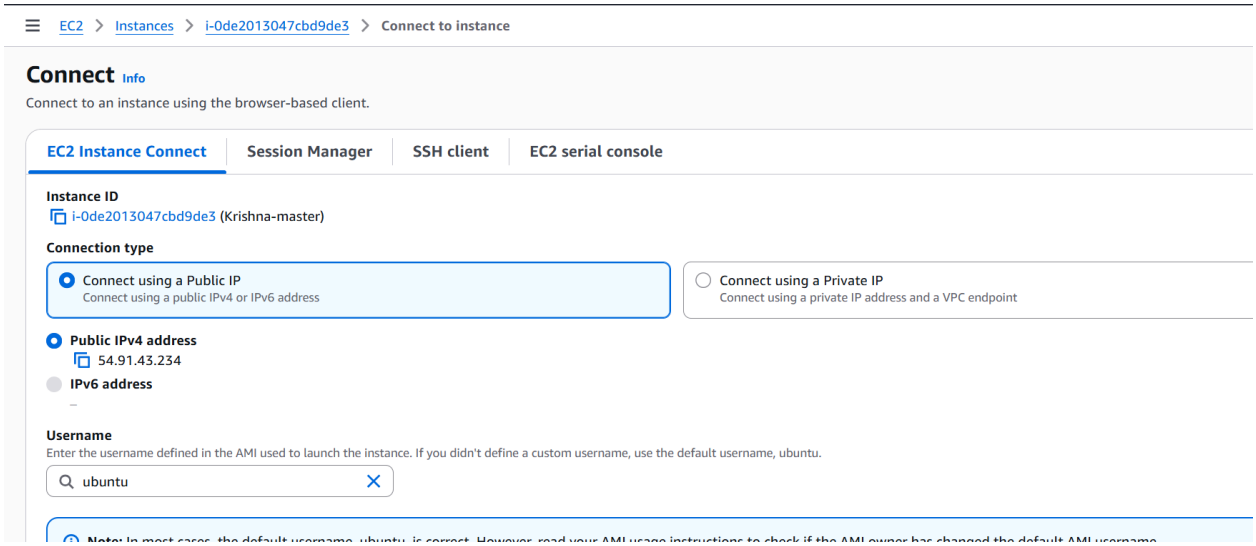
Now you will have 8 policies attached to user




# Step 3. Installations

# a) Connect EC2 to ssh

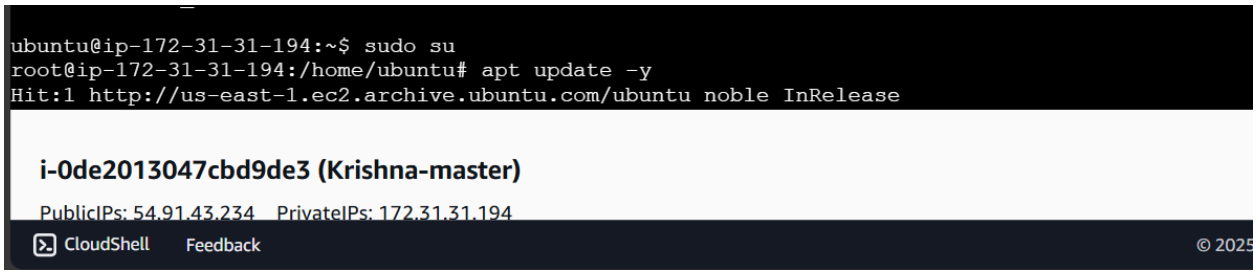





## b) Run basic commands

## sudo su

## apt update -y




## c) Jenkins vi

## Jenkins.sh

**#!/bin/bash**

**# Install OpenJDK 17 JRE Headless**

**sudo apt install openjdk-17-jre-headless -y**

**# Download Jenkins GPG key**

**sudo wget -O /usr/share/keyrings/jenkins-keyring.asc **

**https://pkg.jenkins.io/debian-stable/jenkins.io-2023.key**

**# Add Jenkins repository to package manager sources**

**echo "deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] **

**https://pkg.jenkins.io/debian-stable binary/" | sudo tee **

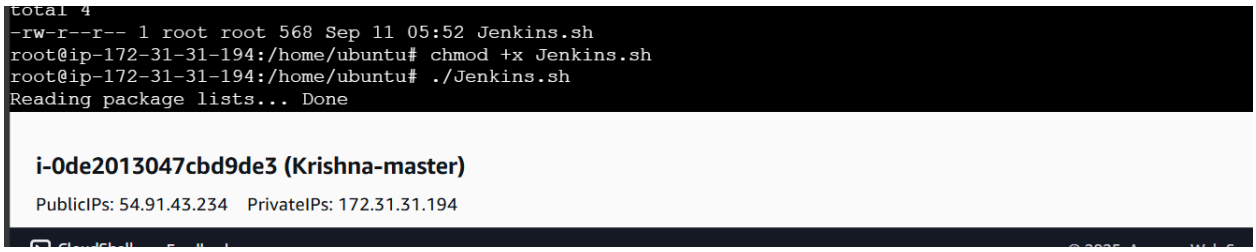
**/etc/apt/sources.list.d/jenkins.list > /dev/null**

**# Update package manager repositories**

**sudo apt-get update**

**# Install Jenkins**

**sudo apt-get install jenkins -y**




**press - esc :wq**

**give permissions and run the file**

**chmod +x jenkin.sh**

**./jenkins.sh**

```
Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

CloudShell Feedback

## Check for java and Jenkins working

## java –version

```
root@ip-172-31-31-194:/home/ubuntu# java --version
openjdk 17.0.16 2025-07-15
OpenJDK Runtime Environment (build 17.0.16+8-Ubuntu-0ubuntu124.04.1)
OpenJDK 64-Bit Server VM (build 17.0.16+8-Ubuntu-0ubuntu124.04.1, mixed mode, sharing)
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

## sudo systemctl status jenkins

```
root@ip-172-31-31-194:/home/ubuntu# sudo systemctl status jenkins
● jenkins.service - Jenkins Continuous Integration Server
     Loaded: loaded (/usr/lib/systemd/system/jenkins.service; enabled; preset: enabled)
     Active: active (running) since Thu 2025-09-11 05:54:34 UTC; 6min ago
   Main PID: 4858 (java)
      Tasks: 45 (limit: 9498)
     Memory: 981.3M (peak: 997.8M)
        CPU: 17.971s
     CGroup: /system.slice/jenkins.service
             └─4858 /usr/bin/java -Djava.awt.headless=true -jar /usr/share/java/jenkins.war --webroot=/var/cache/jenkins/war --httpPort=8080

Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: e18a2d64750f4521883dae1287729e2d
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: This may also be found at: /var/lib/jenkins/secrets/initialAdminPassword
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: ************************************************************
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: ************************************************************
Sep 11 05:54:30 ip-172-31-31-194 jenkins[4858]: ************************************************************
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.268+0000 [id=30]        INFO        jenkins.InitReactorRunner$1#onAttained: Comp
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.284+0000 [id=23]        INFO        hudson.lifecycle.Lifecycle#onReady: Jenkins
Sep 11 05:54:34 ip-172-31-31-194 systemd[1]: Started jenkins.service - Jenkins Continuous Integration Server.
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.441+0000 [id=49]        INFO        h.m.DownloadService$Downloadable#load: Obtai
Sep 11 05:54:34 ip-172-31-31-194 jenkins[4858]: 2025-09-11 05:54:34.451+0000 [id=49]        INFO        hudson.util.Retrier#start: Performed the act
lines 1-20/20 (END)
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

## ✅ Setup Jenkins

**Open Port 8080 in the Jenkins Server**
**Access Jenkins by entering the following in your web browser:**

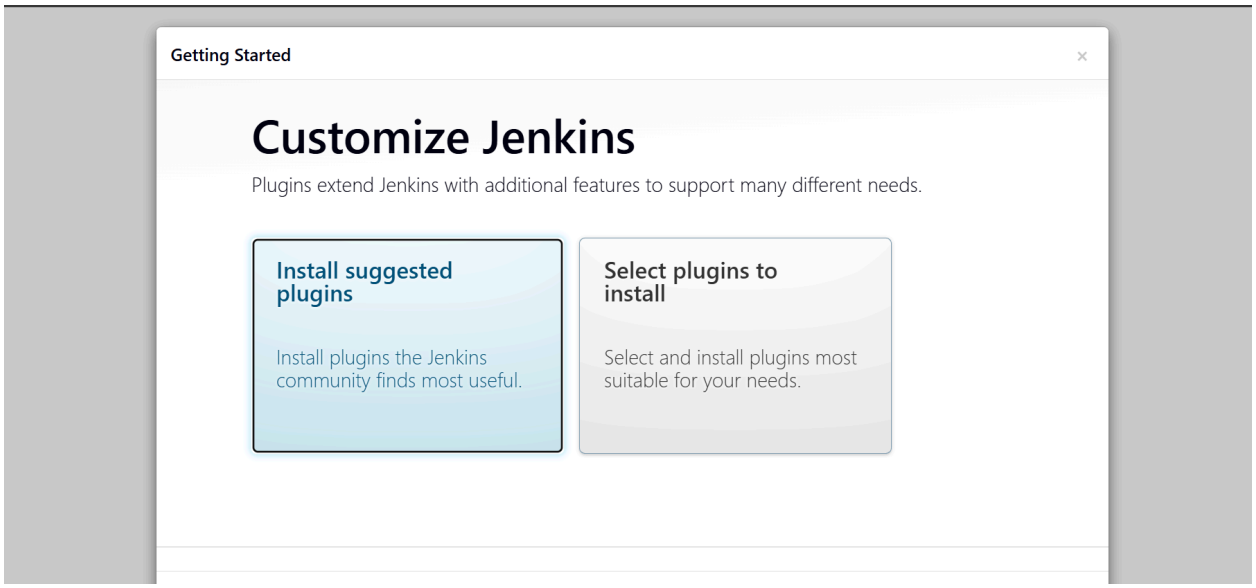
```
http://<ip_address>:8080
```

1. **Replace `<ip_address>` with the public IP address of your Jenkins server.**

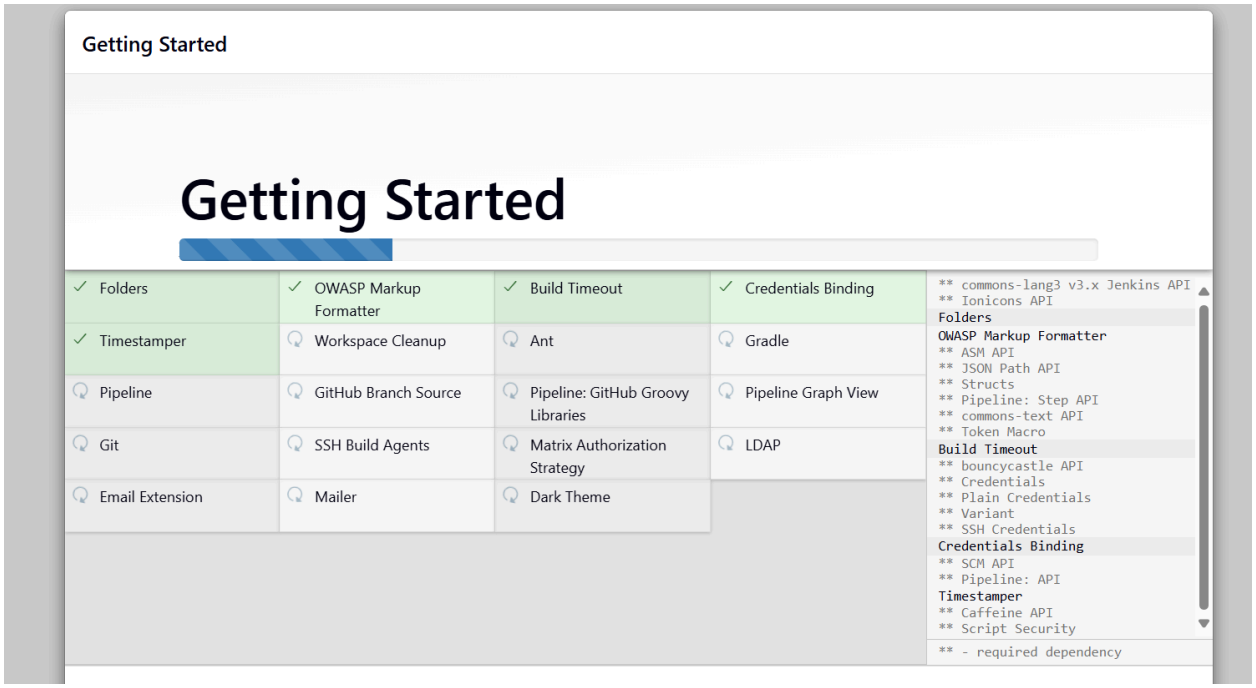
**Retrieve the Initial Administrator Password**
**Run the following command to view the password:**

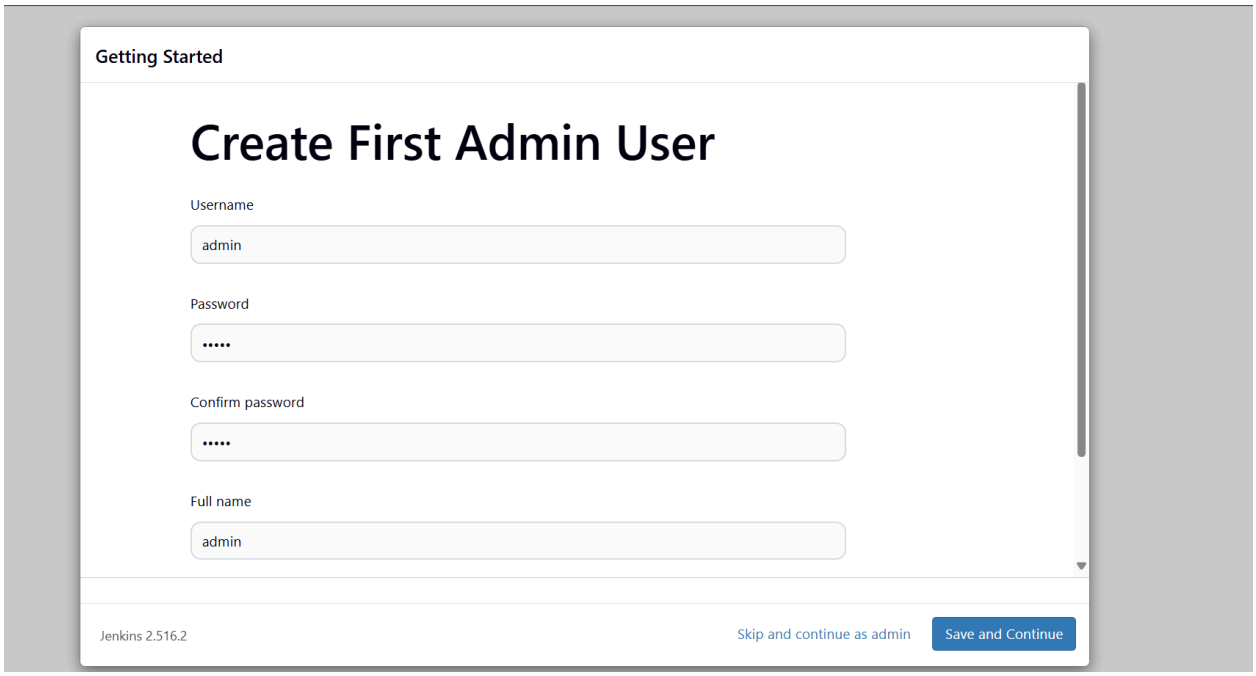
```
cat /var/lib/jenkins/secrets/initialAdminPassword
```

2.
3. **Login to Jenkins**
   **Copy the displayed password and paste it into the Administrator password field on the Jenkins setup page.**




**Click on install all suggested plugins**




## d) Docker

**#!/bin/bash**

**# Update package manager repositories**

```
sudo apt-get update

# Install necessary dependencies
sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key
sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o
/etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources
echo "deb [arch=$(dpkg --print-architecture)
signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories
```

**sudo apt-get update**

**# Install Docker packages**

**sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin**

```
# Install necessary dependencies
sudo apt-get install -y ca-certificates curl

# Create directory for Docker GPG key
sudo install -m 0755 -d /etc/apt/keyrings

# Download Docker's GPG key
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc

# Ensure proper permissions for the key
sudo chmod a+r /etc/apt/keyrings/docker.asc

# Add Docker repository to Apt sources
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc] \
https://download.docker.com/linux/ubuntu \
$(. /etc/os-release && echo \"$VERSION_CODENAME\") stable" | \
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

# Update package manager repositories
sudo apt-get update

# Install Docker packages
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin

-- INSERT --
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

**press – esc :wq**

**give permission and run the file**

**chmod +x docker.sh**

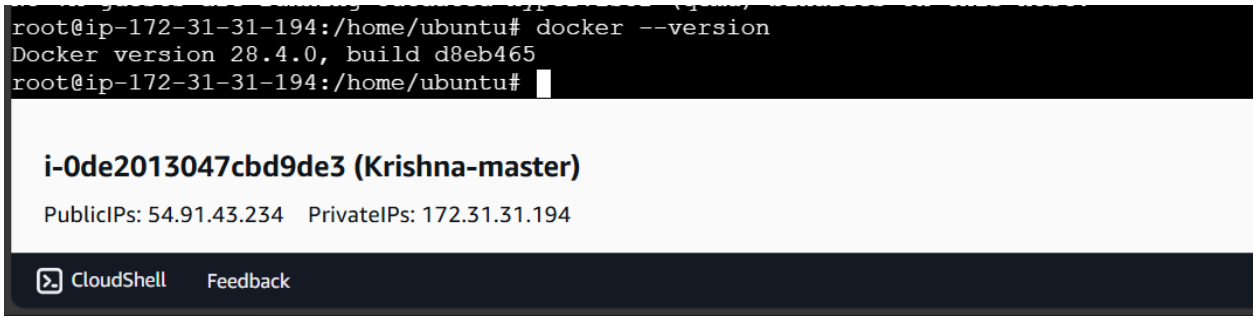
**./docker.sh**

```
root@ip-172-31-31-194:/home/ubuntu# cat /var/lib/jenkins/secrets/initialAdminPassword
e18a2d64750f4521883dae1287729e2d
root@ip-172-31-31-194:/home/ubuntu# vi docker.sh
root@ip-172-31-31-194:/home/ubuntu# chmod +x docker.sh
root@ip-172-31-31-194:/home/ubuntu# ./docker.sh
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

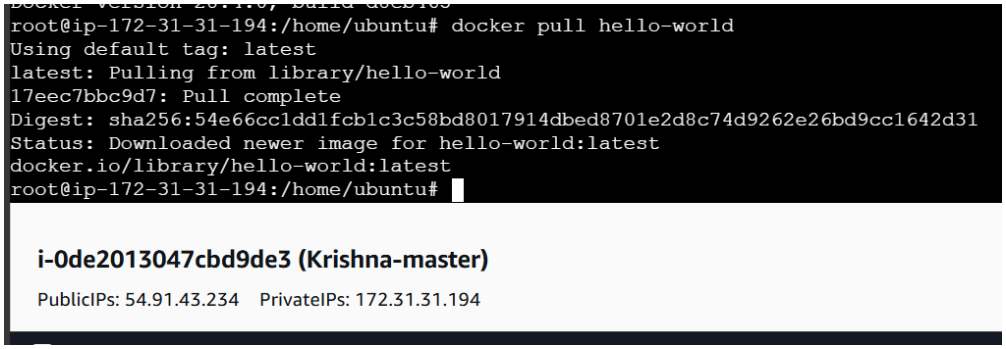
## Docker –version




**Pull a Default Docker Image**

**Pull the `hello-world` image by running the following command:**

```
docker pull hello-world
```

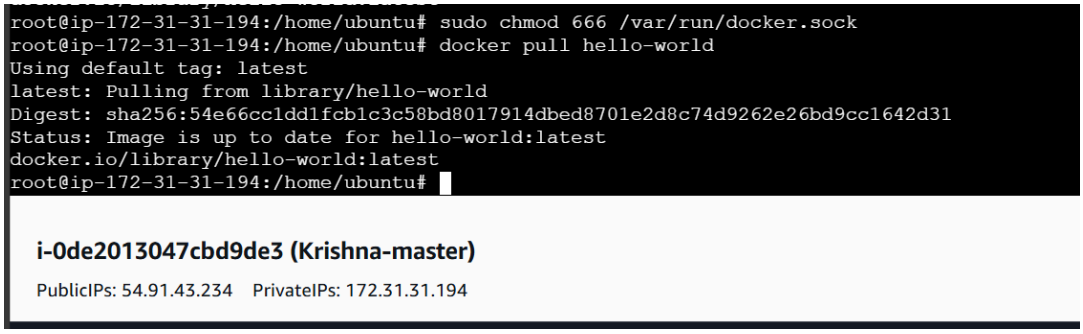


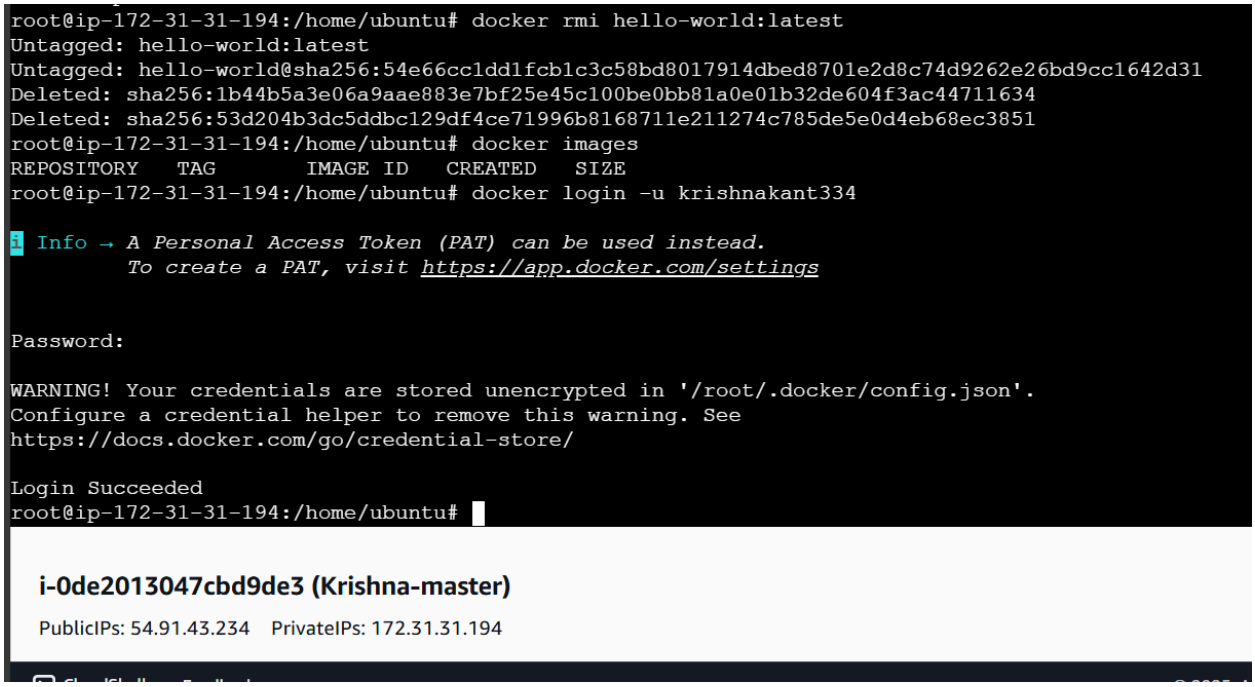

**1.If you see a permission denied error, execute the following command to allow Docker access:**

```
sudo chmod 666 /var/run/docker.sock
```

**2.After changing the permissions, pull the image again:**

```
docker pull hello-world
```






**Remove the Docker Image and Login to Docker Hub**

**Remove the `hello-world` image by running:**

```
docker rmi hello-world:latest
```

1. **Verify that no images remain by running:**

```
docker images
```

2. **➤ The output should show no images.**

3. **Login to Docker Hub**

    - **Open Docker Hub in your browser and log in with your account.**
      **➤ This step is necessary; otherwise, the next steps may result in errors.**

Login from the terminal by running:

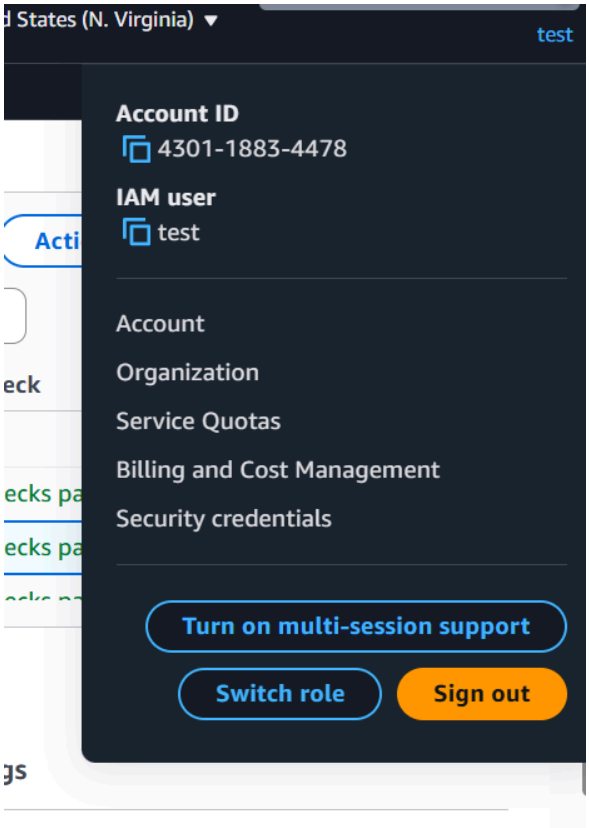
```
docker login -u krishnakant334
```

4. ➢ Press Enter, then type your Docker Hub password and press Enter.

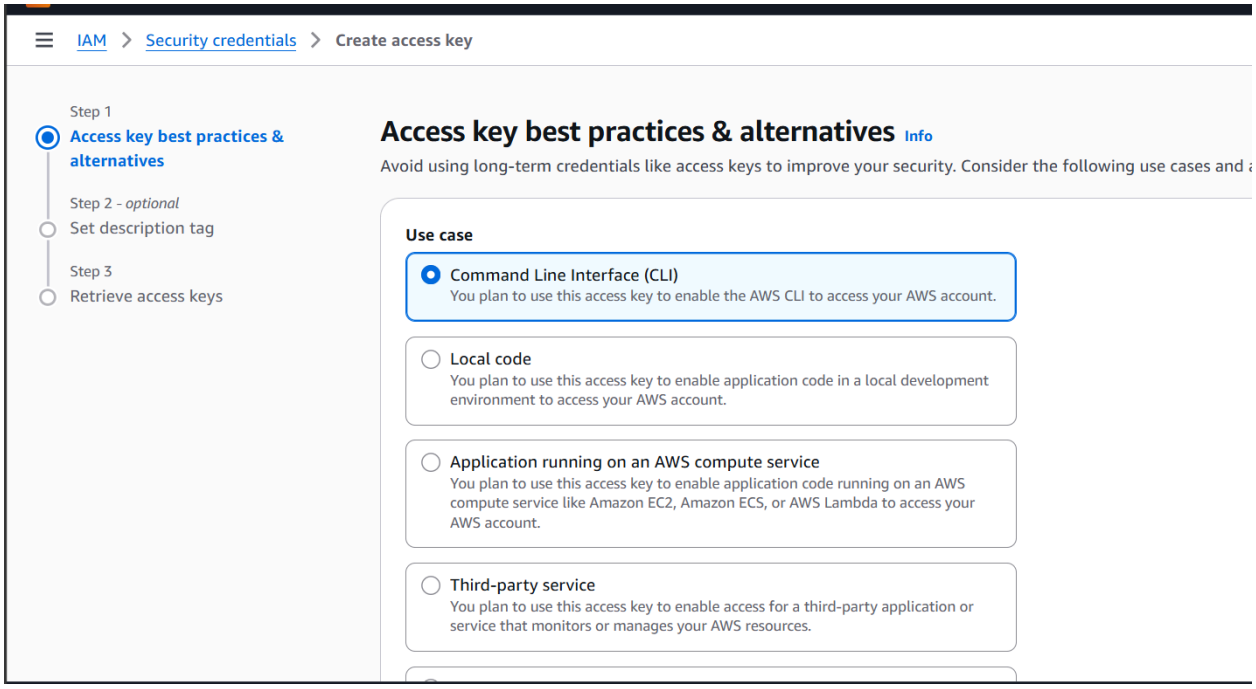
e) AWS cli

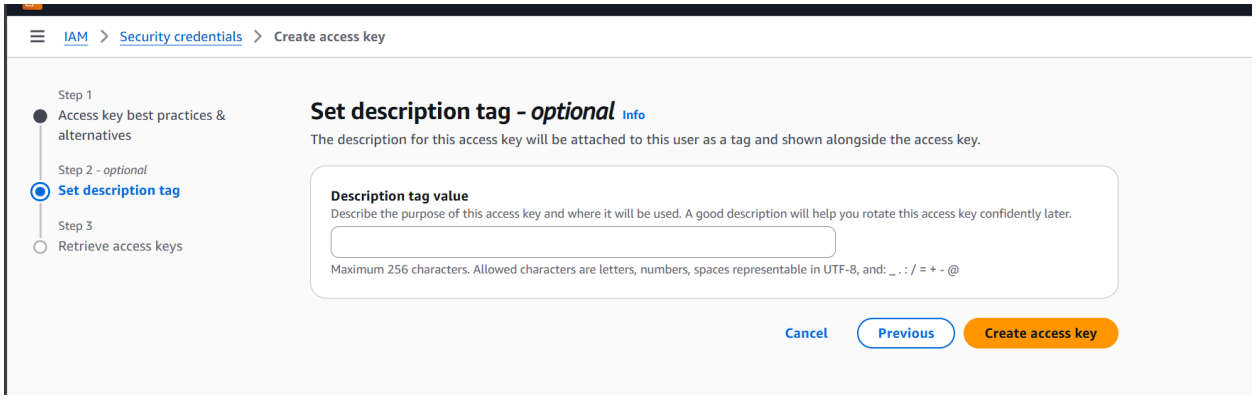
First create Access key and Secret Access key

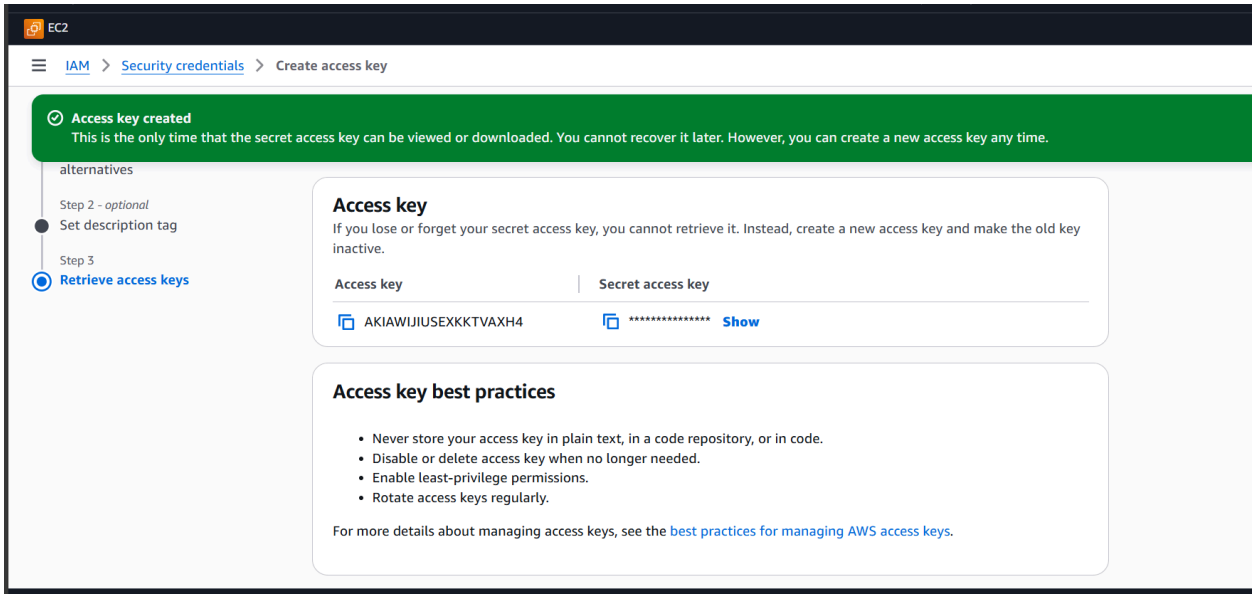
Go to security credentials




Scroll down and click on create access key and create one since I am using sir's account thus I will use the given access key


IAM > Security credentials > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Access key best practices & alternatives Info
Avoid using long-term credentials like access keys to improve your security. Consider the following use cases and
Use case
Command Line Interface (CLI)
You plan to use this access key to enable the AWS CLI to access your AWS account.
Local code
You plan to use this access key to enable application code in a local development environment to access your AWS account.
Application running on an AWS compute service
You plan to use this access key to enable application code running on an AWS compute service like Amazon EC2, Amazon ECS, or AWS Lambda to access your AWS account.
Third-party service
You plan to use this access key to enable access for a third-party application or service that monitors or manages your AWS resources.


IAM > Security credentials > Create access key
Step 1
Access key best practices & alternatives
Step 2 - optional
Set description tag
Step 3
Retrieve access keys
Set description tag - optional Info
The description for this access key will be attached to this user as a tag and shown alongside the access key.
Description tag value
Describe the purpose of this access key and where it will be used. A good description will help you rotate this access key confidently later.
Maximum 256 characters. Allowed characters are letters, numbers, spaces representable in UTF-8, and: _ . : / = + - @
Cancel
Previous
Create access key




**Install AWS CLI on the Server**

**Create the script file by running:**

```
vi aws.sh
```

**1.Add the following commands inside the file:**

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"

sudo apt install unzip -y

unzip awscliv2.zip

sudo ./aws/install
```

**2.Save and exit the file:**

- **Press Esc, type :wq, and press Enter.**

2.Give execute permissions to the script:

```
chmod +x aws.sh
```

3.Run the script:

```
./aws.sh
```

## Install AWS CLI using the downloaded package

Download the AWS CLI installer:

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o "awscliv2.zip"
```

1. Install `unzip` if it's not already installed:

```
sudo apt install unzip -y
```

2. Extract the AWS CLI installer:

```
unzip awscliv2.zip
```

3. 
4. Run the installation script:

```
sudo ./aws/install
```

```
Login Succeeded
root@ip-172-31-31-194:/home/ubuntu# vi aws.sh
root@ip-172-31-31-194:/home/ubuntu#
```

i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

CloudShell Feedback

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
"awscliv2.zip"
sudo apt install unzip
unzip awscliv2.zip
sudo ./aws/install
~
~
~
~
~
~
~
~
~
```

```
https://docs.docker.com/go/credential-store/

Login Succeeded
root@ip-172-31-31-194:/home/ubuntu# vi aws.sh
root@ip-172-31-31-194:/home/ubuntu# ^[[200~chmod +x aws.sh ~^C
root@ip-172-31-31-194:/home/ubuntu# chmod +x aws.sh
root@ip-172-31-31-194:/home/ubuntu# ./aws.sh
curl: option -o: requires parameter
```

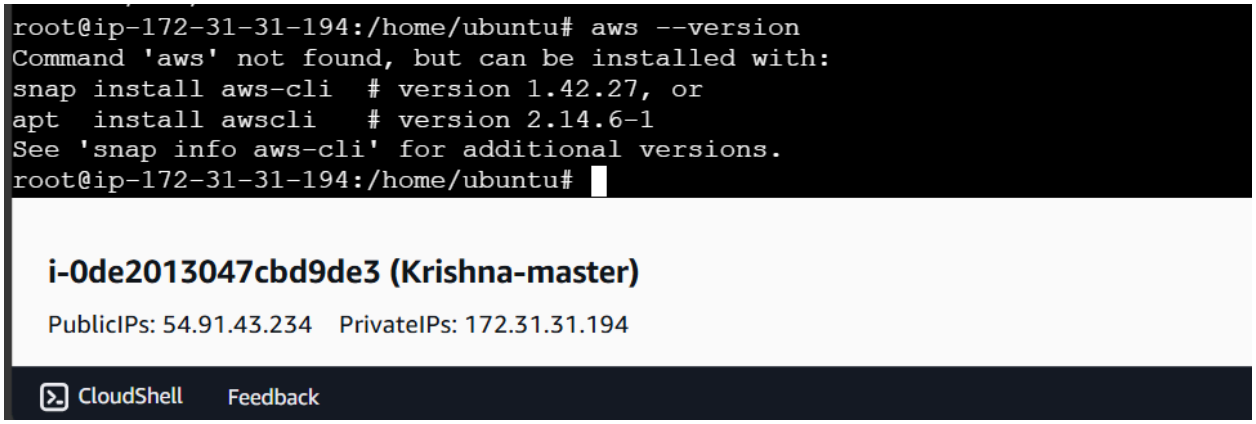
i-0de2013047cbd9de3 (Krishna-master)

PublicIPs: 54.91.43.234 PrivateIPs: 172.31.31.194

CloudShell Feedback

**Aws version check**

**aws –version**



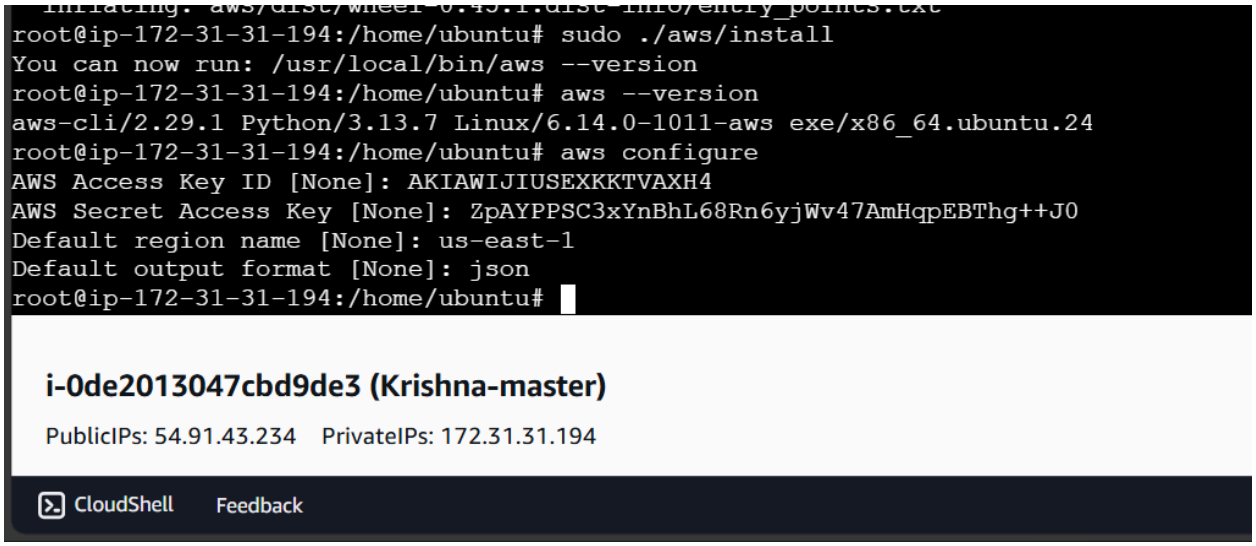

**Configure AWS CLI**

**Run the following command to start AWS configuration:**

```
aws configure
```

**Enter the requested details:**

- **AWS Access Key ID:** ***Enter your access key***
- **AWS Secret Access Key:** ***Enter your secret key***
- **Default region name:** `<your_working_region>`
- **Default output format:** `json`




## Install kubectl

**Create the script file by running:**

```
vi kubectl.sh
```

1.

**Add the following commands inside the file:**

```
curl -o kubectl
https://amazon-eks.s3.us-west-2.amazonaws.com/1.19.6/2021-01-05/bin/linux/amd64/kubectl

chmod +x ./kubectl

sudo mv ./kubectl /usr/local/bin

kubectl version --short --client
```

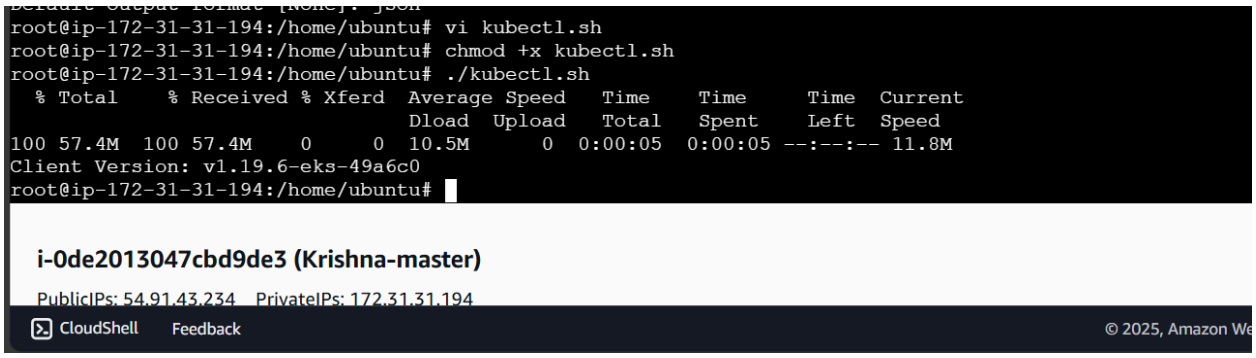
2.
3. **Save and exit the file:**
    - **Press `Esc`, type `:wq`, and press `Enter`.**

**Give execute permissions to the script:**

```
chmod +x kubectl.sh
```

4.
5. **Run the script:**

```
./kubectl.sh
```




# Install eksctl

**Create the script file by running:**

```
vi eksctl.sh
```

1.

**Add the following commands inside the file:**

```
curl --silent --location
"https://github.com/weaveworks/eksctl/releases/latest/download/eksctl_$(uname
-s)_amd64.tar.gz" | tar xz -C /tmp

sudo mv /tmp/eksctl /usr/local/bin

eksctl version
```

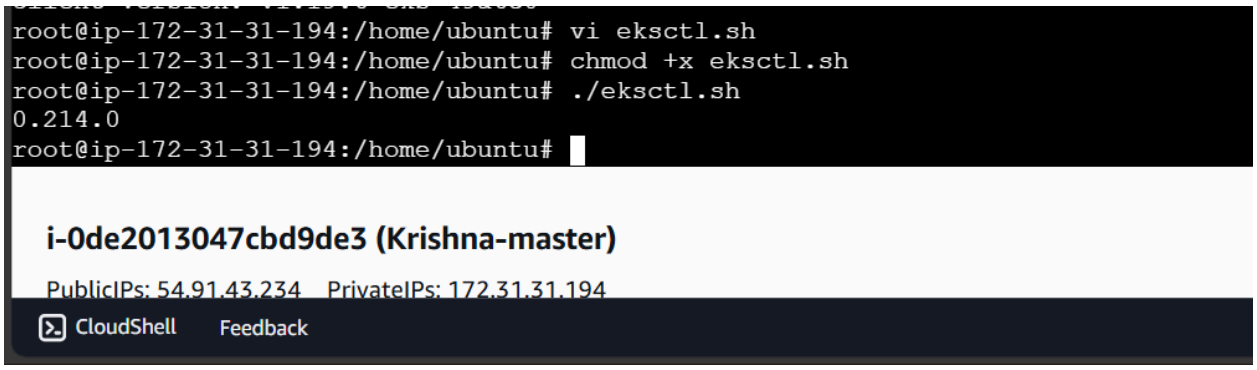
2.
3. **Save and exit the file:**
    - **Press Esc, type :wq, and press Enter.**

**Give execute permissions to the script:**

```
chmod +x eksctl.sh
```

4.
5. **Run the script:**

```
./eksctl.sh
```




## Install Ansible

Create the script file by running:

```
vi ansible.sh
```

1.

Add the following commands inside the file:

```
sudo add-apt-repository --yes --update ppa:ansible/ansible
sudo apt install ansible -y
```

2.
3. Save and exit the file:
    - Press Esc, type :wq, and press Enter.

Give execute permissions to the script:

```
chmod +x ansible.sh
```

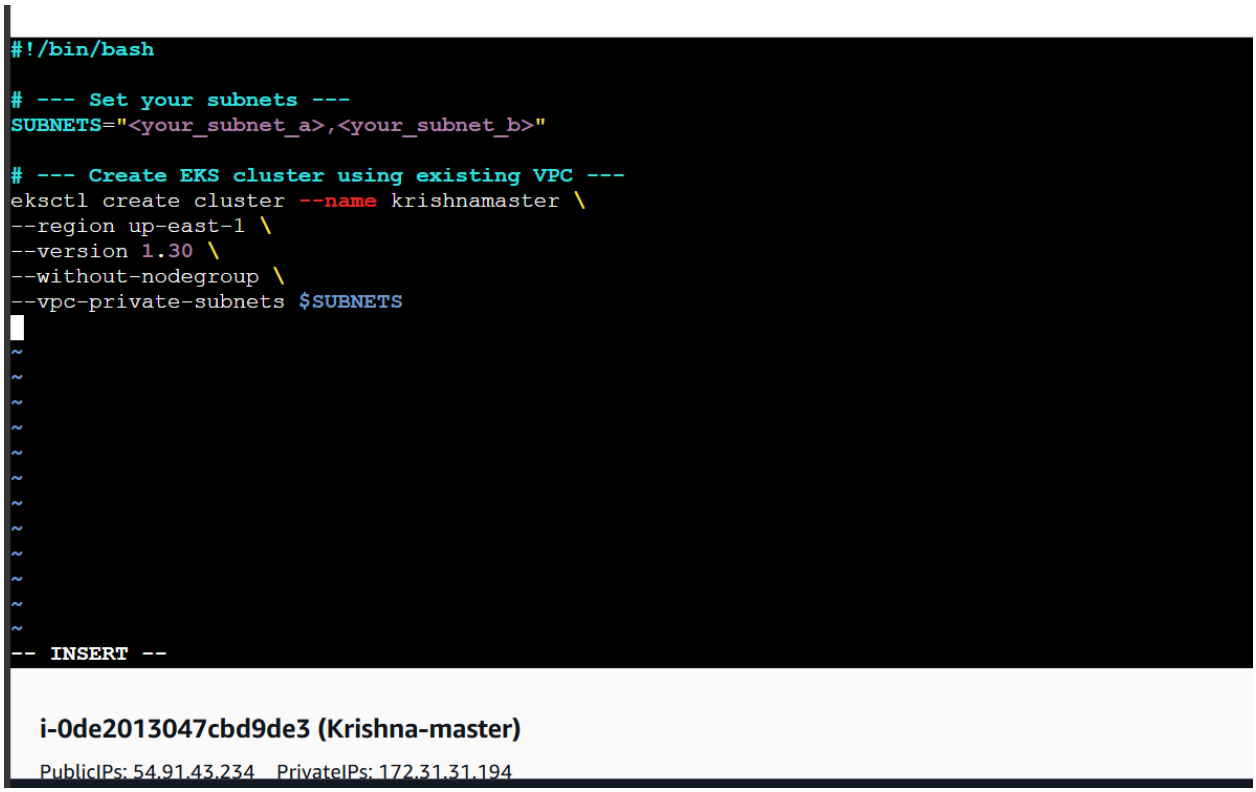
4.

Run the script:

```
./ansible.sh
```

5.
6. Check the installed version of Ansible:

```
ansible --version
```




```
#!/bin/bash


# --- Set your subnets ---

SUBNETS="subnet-09527e09d2b636418,subnet-01dfc86302e426722"


# --- Create EKS cluster using existing VPC ---

eksctl create cluster --name krishnamaster \

--region us-east-1 \

--version 1.30 \

--without-nodegroup \

--vpc-private-subnets $SUBNETS
```

## Instructions

Save this script to cluster.sh:

```
vi cluster.sh
```

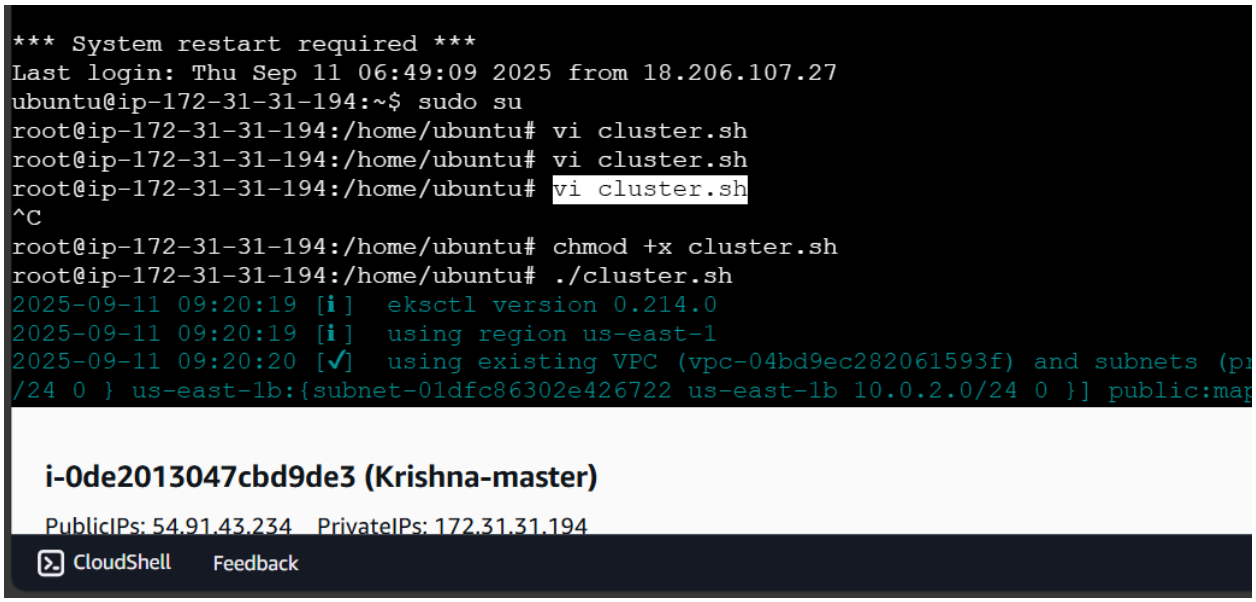
1. Paste the above content.
2. Save and exit (Esc, type :wq, press Enter).

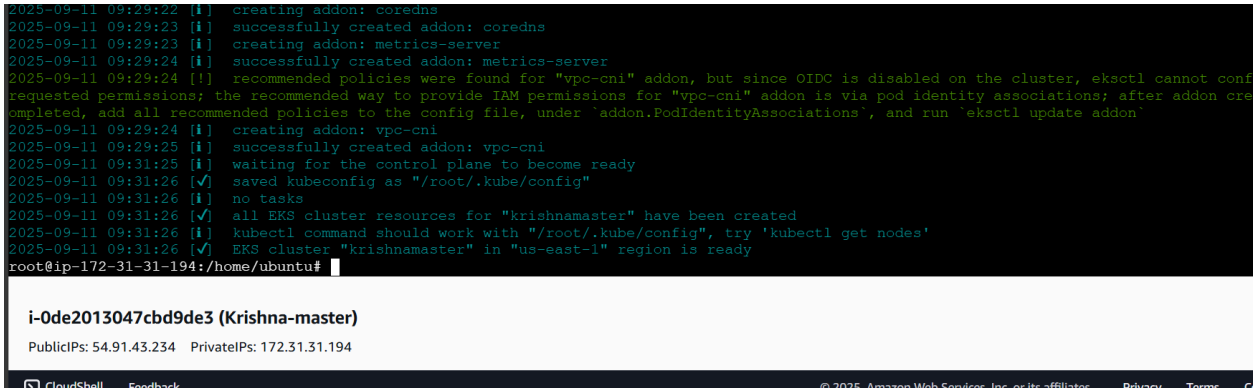
Make the script executable:

```
chmod +x cluster.sh
```

3.
4. Run the script:

```
./cluster.sh
```






## b) Associate IAM OIDC Provider

**1.Open the script file:**

```
vi ekassociation.sh
```

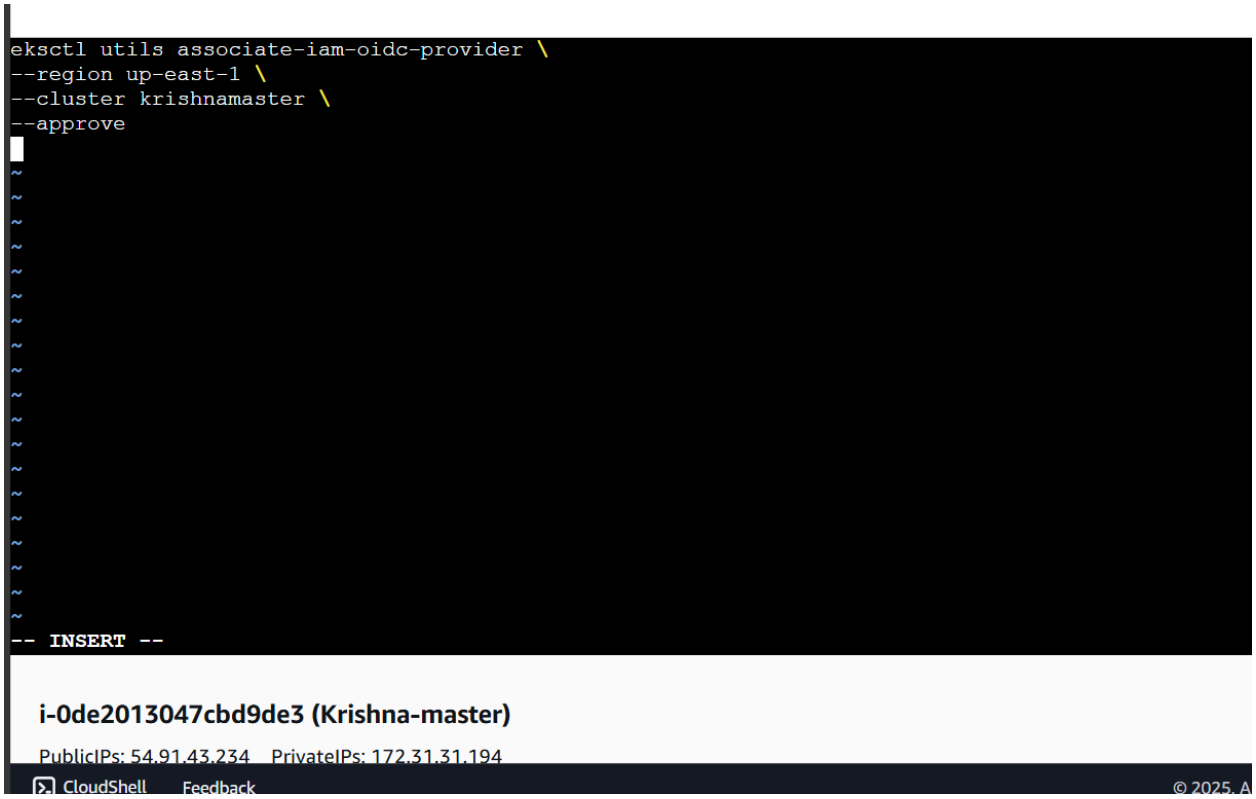
**2.Add the following content inside the file:**

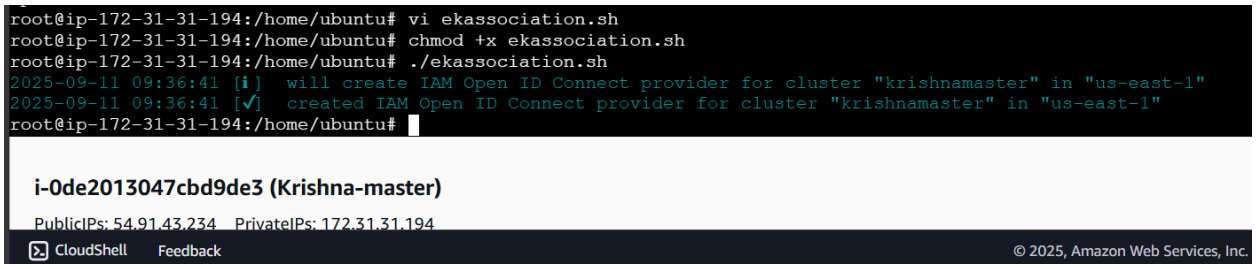
```
eksctl utils associate-iam-oidc-provider \
--region up-east-1 \
--cluster krishnamaster \
--approve
```

**3.Save and exit the file.**

**4.Give execute permission and run the script:**

```
chmod +x ekassociation.sh
./ekassociation.sh
```




```
#!/bin/bash


eksctl create nodegroup \

--cluster krishnamaster \

--region us-east-1 \

--name node2 \

--node-type t3.medium \

--nodes 2 \
```

```
--nodes-min 1 \

--nodes-max 2 \

--node-volume-size 20 \

--ssh-access \

--ssh-public-key <your_key_name> \

--managed \

--asg-access \

--external-dns-access \

--full-ecr-access \

--appmesh-access \

--alb-ingress-access \

--node-private-networking \

--subnet-ids subnet-09527e09d2b636418,subnet-01dfc86302e426722 \

--tags "Environment=dev"
```

## instructions

Save this script as `createnode.sh`:

```
vi createnode.sh
```

1. Paste the content above, then save and exit (`Esc`, `:wq`, `Enter`).

Make it executable:

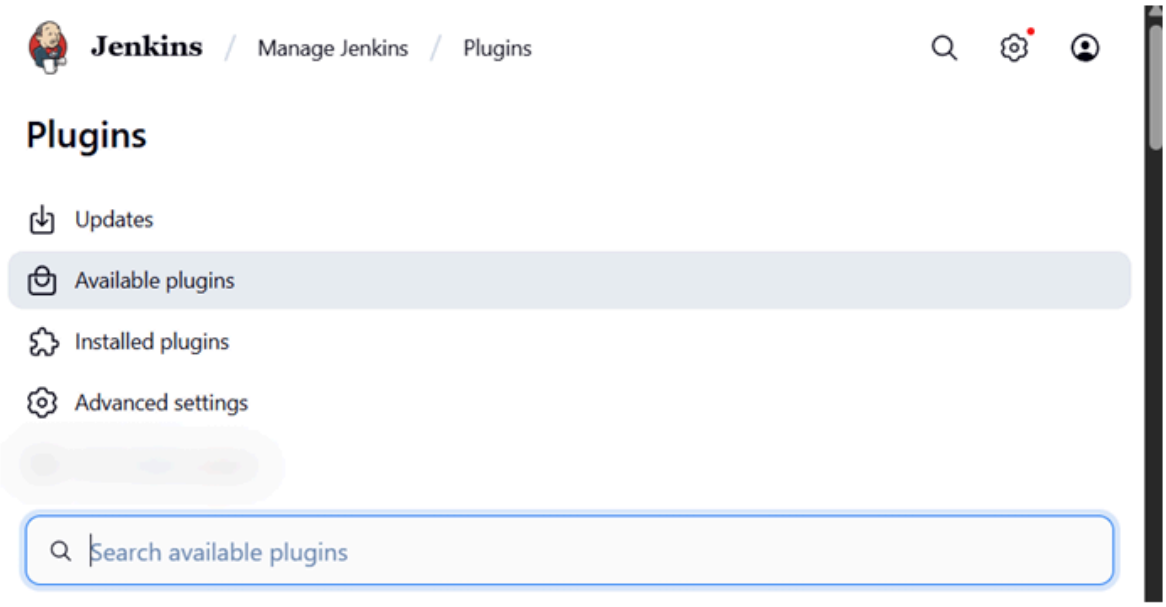
```
chmod +x createnode.sh
```

2.
3. Run the script:

```
./createnode.sh
```

# Step 5. Establishing Jenkins Plugins

a) Open Jenkins -> Manage Jenkins -> Plugins -> Available Plugins



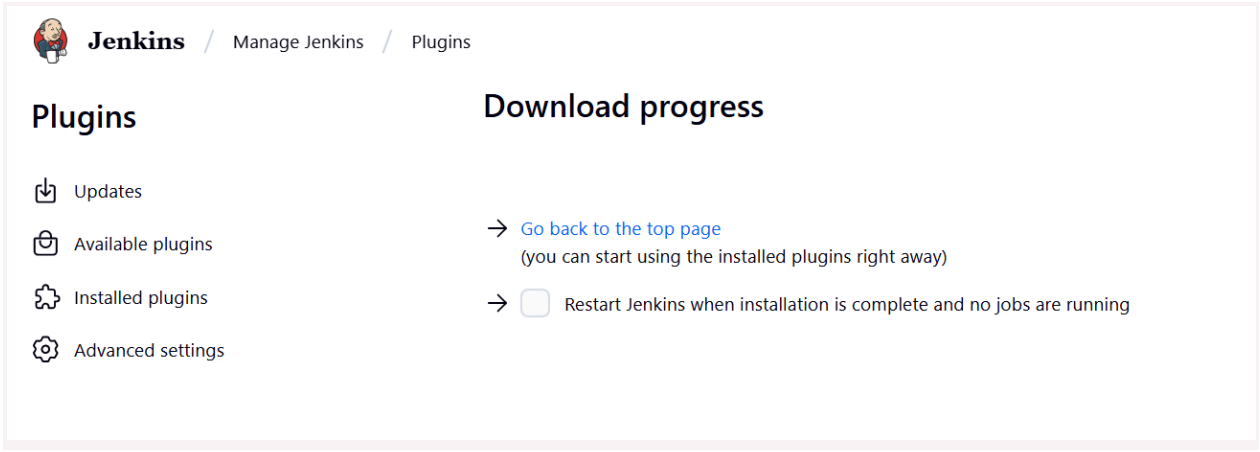
Install all 13 Plugins listed below and after that restart Jenkins

1. Pipeline: Stage View
2. Docker
3. Docker Commons
4. Docker Pipeline
5. Docker API
6. docker-build-step
7. Kubernetes
8. Kubernetes Client API
9. Kubernetes Credentials
10. Kubernetes CLI
11. Config File Provider
12. Eclipse Temurin Installer (for Java)
13. Ansible

Jenkins is restarting

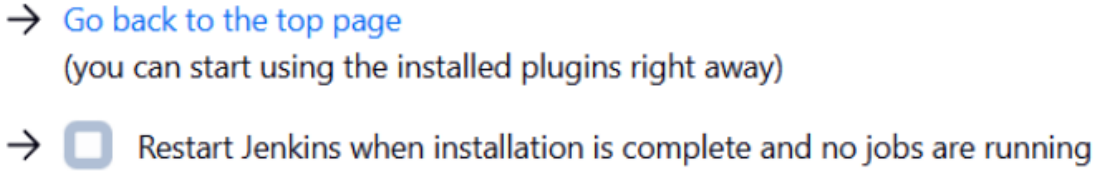
Your browser will reload automatically when Jenkins is ready.

Safe Restart
Builds on agents can usually continue.

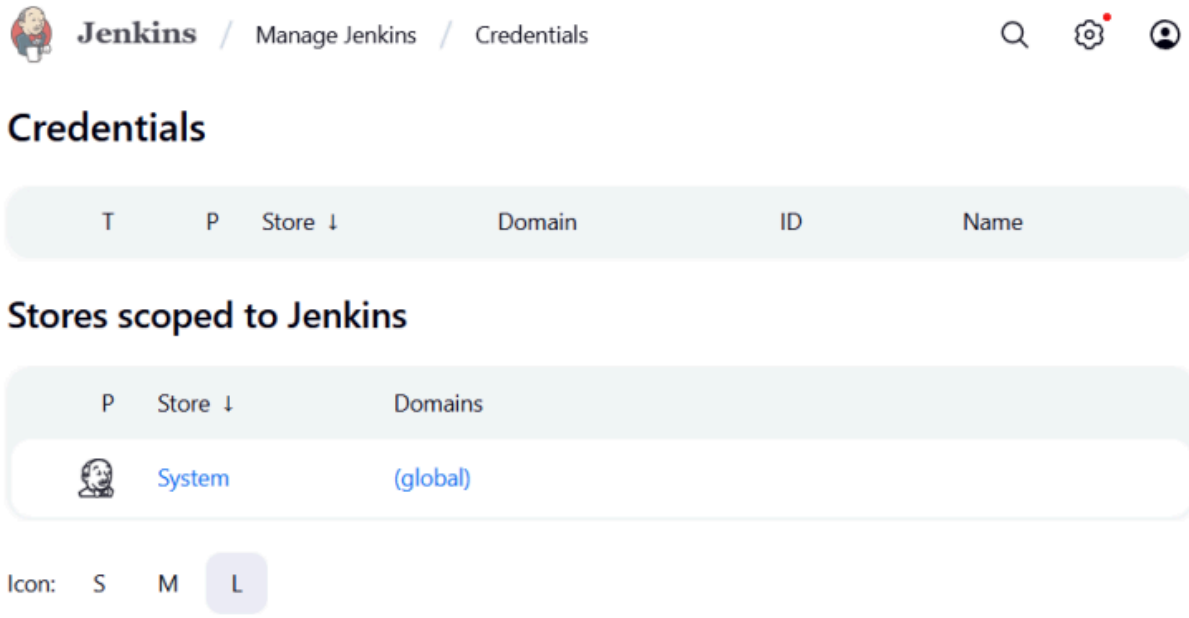

Jenkins / Manage Jenkins / Plugins
Plugins
Updates
Available plugins
Installed plugins
Advanced settings
Download progress
Go back to the top page
(you can start using the installed plugins right away)
Restart Jenkins when installation is complete and no jobs are running

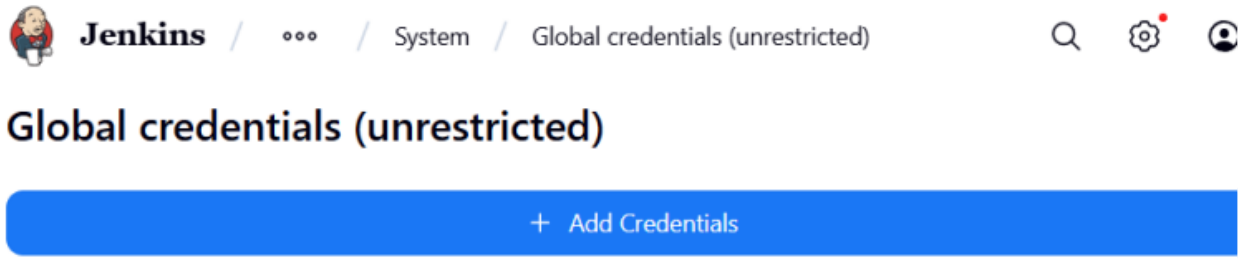
| | |
|---|---|
| Preparation | • Checking internet connectivity<br>• Checking update center connectivity<br>• Success |
| commons-lang3 v3.x Jenkins API | Success |
| Ionicons API | Success |
| Folders | Success |
| OWASP Markup Formatter | Success |
| ASM API | Success |
| JSON Path API | Success |
| Structs | Success |
| Pipeline: Step API | Success |
| commons-text API | Success |
| Token Macro | Success |
| Build Timeout | Success |
| bouncycastle API | Success |
| Credentials | Success |
| Plain Credentials | Success |
| Variant | Success |
| SSH Credentials | Success |
| Credentials Binding | Success |
| SCM API | Success |
| Pipeline: API | Success |
| Timestamper | Success |
| Caffeine API | Success |

**after installation check on Restart option**




**b) Go to Manage jenikns -> Credentials (Under security) ->click global -> Add credentials**




**Make below changes**

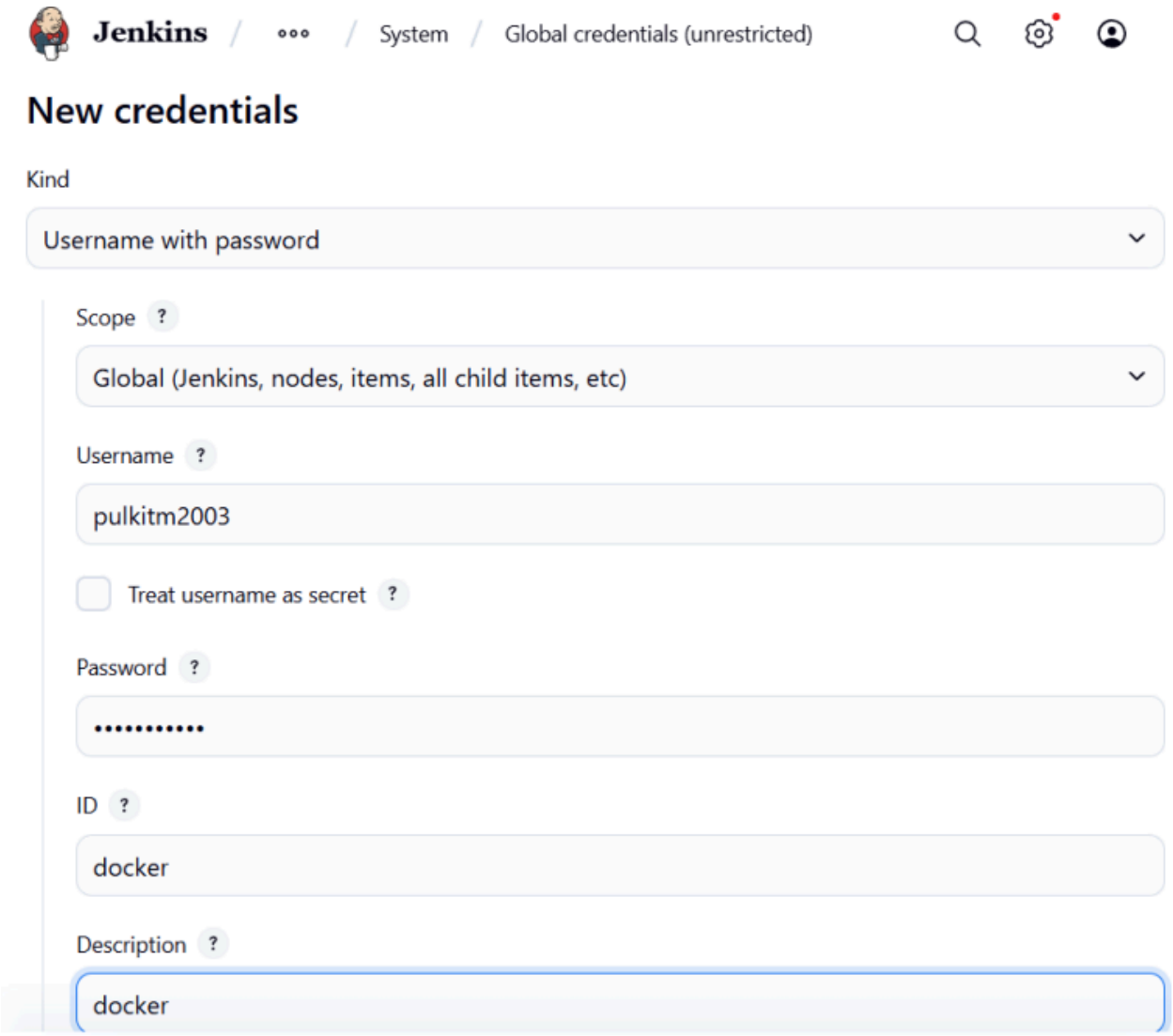
**User name = your docker username**

**Password = your docker hub password**

**Id = docker**

**Description = docker**

**Click create**




**Add global credentials for ansible too**

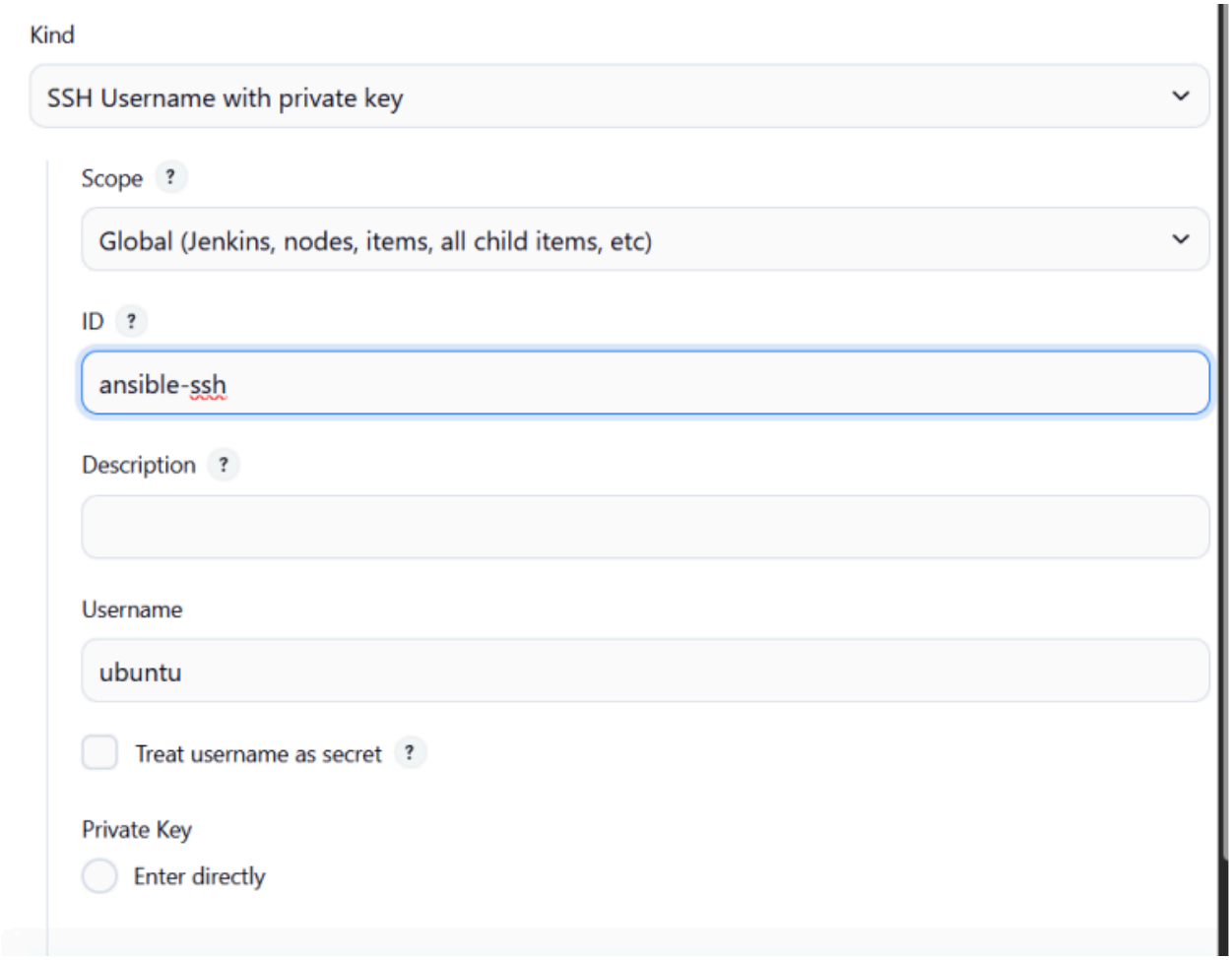
**Kind : SSH Username with private key**

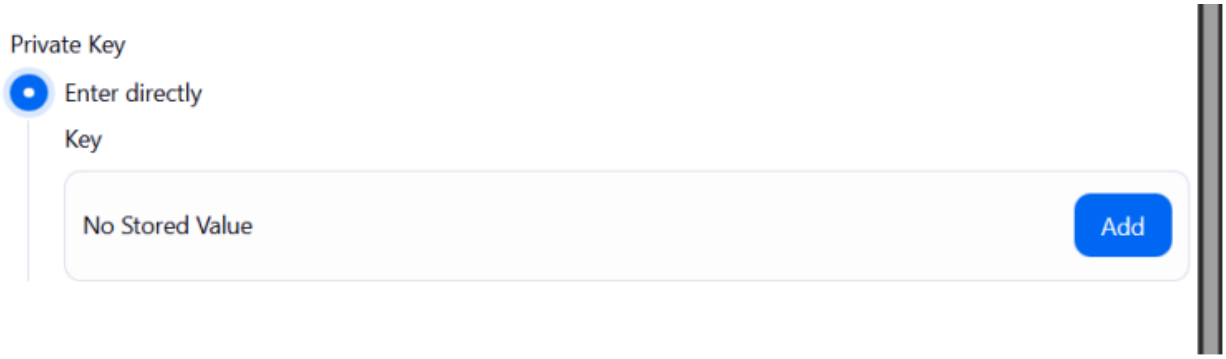
**Scope : Global**

**ID : ansible-ssh**

**Username : ubuntu**

**Click on Private key enter directly and enter the private key used to make master ec2**





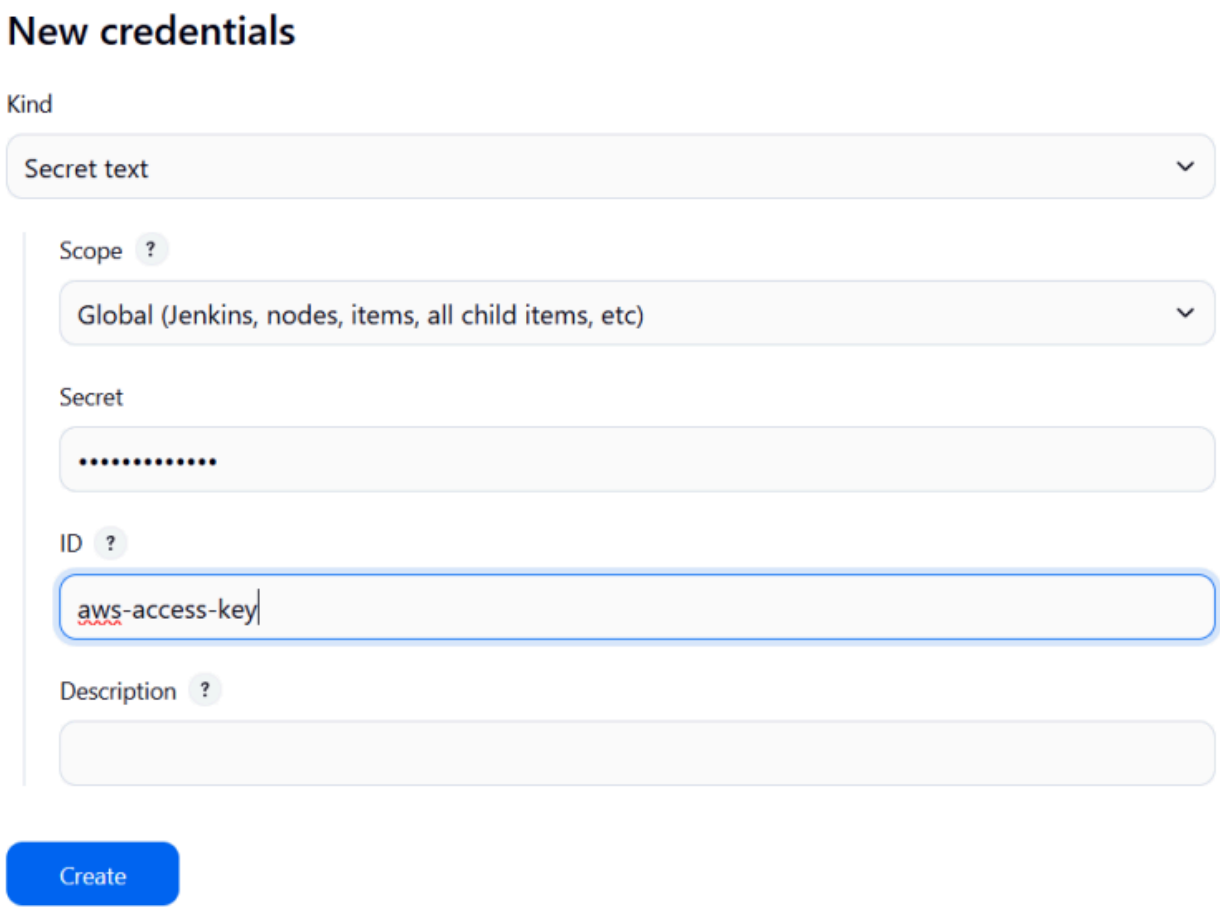

**Add Credentials for AWS access key**

**Kind : Secret key**

**Scope : Global**

**Secret : <add_accesskey>**

**ID : aws-access-key**



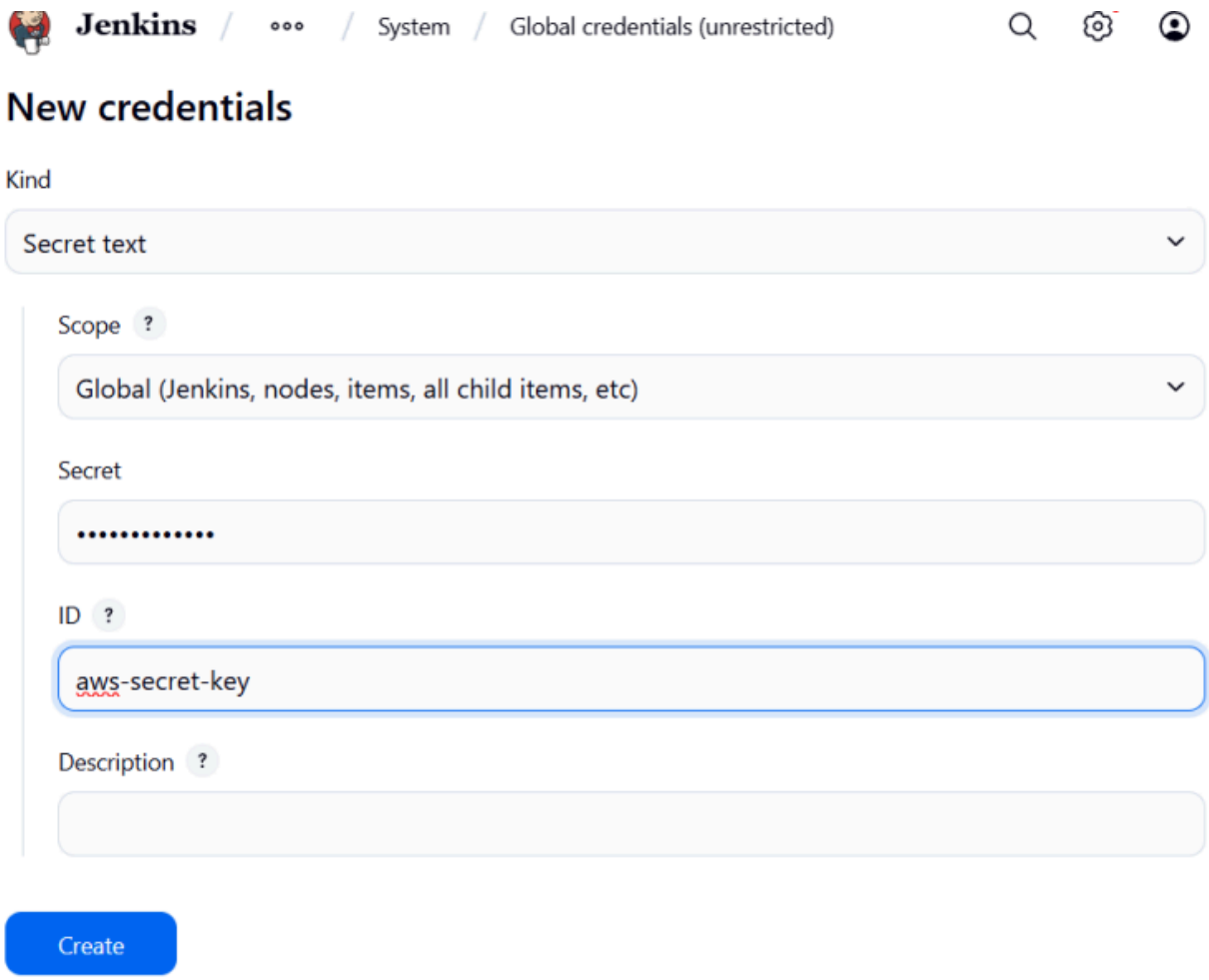

**Add Credentials for AWS Secret access key**

**Kind : Secret key**

**Scope : Global**

**Secret : <add_Secret_accesskey>**

**ID : aws-secret-key**




**Click on create**

| T | P | Store ↓ | Domain | ID | Name |
|---|---|---|---|---|---|
| | | System | (global) | ansible-ssh | ubuntu |
| | | System | (global) | aws-access-key | aws-access-key |
| | | System | (global) | aws-secret-key | aws-secret-key |
| | | System | (global) | dockerhub-cred-id | pulkitm2003/****** (dockerhub-cred-id) |

Stores scoped to Jenkins

**c) go to -> Manage Jenkins -> Tools (under system configuration) ->**

**click on Add jdk**
**name: jdk17**
**select install automatically**

**under installer dropdown**
**select “install from adoptium.net”**
**version? : select jdk-17.0.8.1+1 version**
**----------**
**----------**
**add docker**
**name: docker**

**select install automatically**
**add installer**
**Download from docker.com**

---

add ansible
name: ansible

leave as it is
---

click on apply and save

# Step 6. Creating Dynamic Inventory

a) Give different names

since all our nodes have same names thus we need to give them different names to differentiate here we will use the tag we gave earlier Environment = dev

vi tags.sh

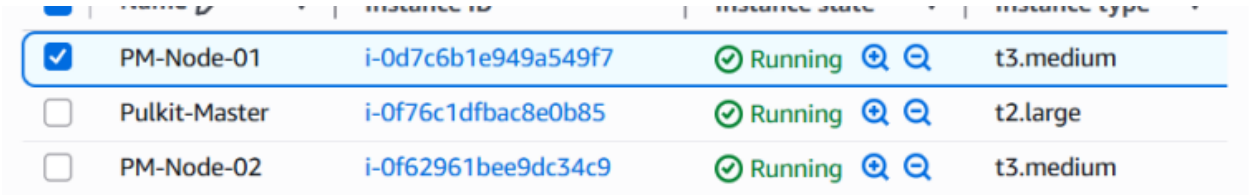
```
# Fetch instance IDs that match Environment=dev and Role=web
instance_ids=$(aws ec2 describe-instances \
  --filters "Name=tag:Environment,Values=dev" "Name=instance-state-name,Values=running" \
  --query 'Reservations[*].Instances[*].InstanceId' \
  --output text)

# Sort instance IDs deterministically
sorted_ids=($(echo "$instance_ids" | tr '\t' '\n' | sort))

# Rename instances sequentially
counter=1
for id in "${sorted_ids[@]}"; do
  name="<name_you_want>-$(printf "%02d" $counter)"
  echo "Tagging $id as $name"
```

```
  aws ec2 create-tags --resources "$id" \
    --tags Key=Name,Value="$name"
  ((counter++))
done
```

**press esc:wq**

**give permission and run file**

```
root@ip-172-31-4-83:/home/ubuntu# chmod +x tags.sh
root@ip-172-31-4-83:/home/ubuntu# ./tags.sh
Tagging i-0d7c6b1e949a549f7 as PM-Node-01
Tagging i-0f62961bee9dc34c9 as PM-Node-02
```



| | | | | |
|---|---|---|---|---|
| ☑ | PM-Node-01 | i-0d7c6b1e949a549f7 | Running | t3.medium |
| ☐ | Pulkit-Master | i-0f76c1dfbac8e0b85 | Running | t2.large |
| ☐ | PM-Node-02 | i-0f62961bee9dc34c9 | Running | t3.medium |

**b) Create Ansible key**

```
ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
```

**press enter 3 times**

```
root@ip-172-31-4-83:/home/ubuntu# ssh-keygen -t rsa -b 4096 -C "Ansible-Master"
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa
Your public key has been saved in /root/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:czktc4W2m5zxI6BqOQurUbumtL25mPwGLihJeXCDUpM Ansible-Master
The key's randomart image is:
+---[RSA 4096]----+
|   .             |
|  E          .   |
| ...        o .  |
|.o o       + o   |
|. + o   S B =    |
| o.o .   + B *   |
|oo+.o  ..   * o  |
|=o.*o=+.     . . |
|..B*O+oo         |
+----[SHA256]-----+
root@ip-172-31-4-83:/home/ubuntu#
```

## c) Creating Yaml file that will hold IP address of node

```
mkdir inventory
```

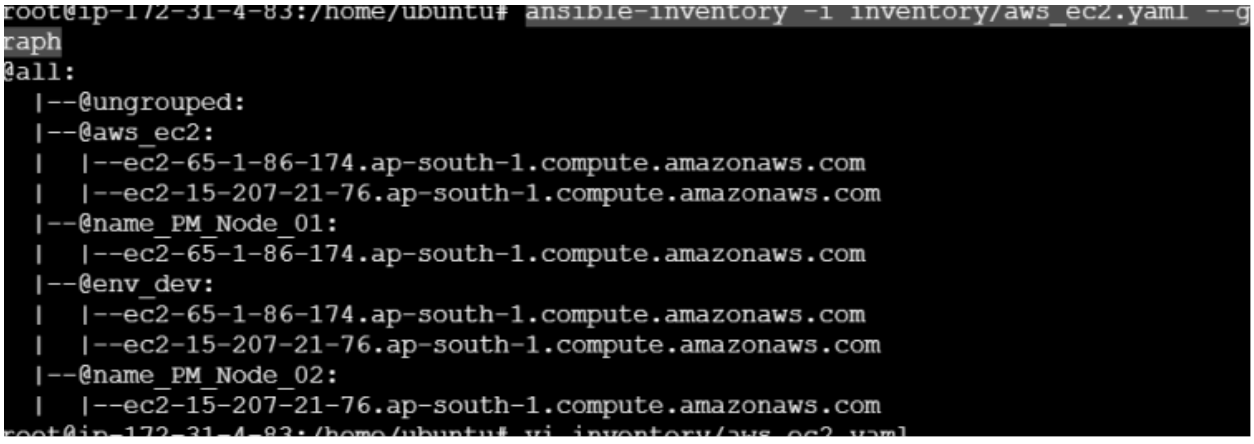
```
vi inventory/aws_ec2.yaml
```

```
plugin: amazon.aws.aws_ec2
regions:
   - <your_region_name>
filters:
  tag:Environment: dev
  instance-state-name: running
```

```
compose:
  ansible_host: public_ip_address
keyed_groups:
  - key: tags.Name
    prefix: name
  - key: tags.Environment
    prefix: env
```

to test run this command

```
ansible-inventory -i inventory/aws_ec2.yaml
-graph
```




# Step 7. Connecting Ansible Master - Node

**a) Creating Ansible Environment**

We are setting up separate environment to isolate Ansible dependencies

Install venv module if not already present :

```
sudo apt install python3-venv -y
```

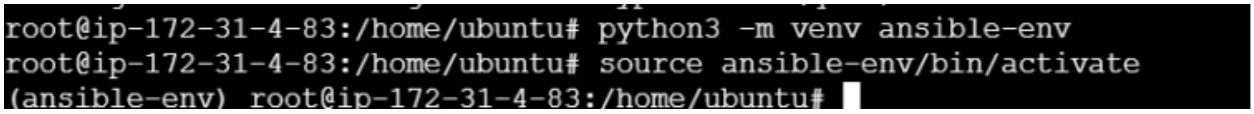
Create a virtual environment :

```
python3 -m venv ansible-env
```

Activate it :

```
source ansible-env/bin/activate
```

we can see now we have entered into ansible environment




Install required Python packages :

```
pip install boto3 botocore docker
```

**b) Creating Key file**

Open the key pair file in notepad and copy its content


**Paste this content in file below**

**vi Krishna-New-kp.pem**

**(name should be same as key pair used to create master and nodes )**

**After pasting key value here press – esc:wq**

**Give read only permission**

**Chmod 400 Krishna-New-kp.pem**

**c) Give over ride permission**

**since while adding key in nodes nodes require permission as yes but we cannot open each node and give it permission as there are**

multiple nodes thus we will over ride those permissions

```
vi ansible.cfg

[defaults]

inventory = ./inventory/aws_ec2.yaml

host_key_checking = False


[ssh_connection]

ssh_args = -o StrictHostKeyChecking=no -o UserKnownHostsFile=/dev/null
```

d) Script to connect Nodes and Master

```
vi copy-public-key.sh

#!/bin/bash


# Define vars

PEM_FILE="Krishna-New-kp.pem"

PUB_KEY=$(cat ~/.ssh/id_rsa.pub)
```

```
USER="ec2-user"  # Since nodegroup is Linux
INVENTORY_FILE="inventory/aws_ec2.yaml"

# Extract hostnames/IPs from dynamic inventory
HOSTS=$(ansible-inventory -i $INVENTORY_FILE --list | jq -r '._meta.hostvars | keys[]')

for HOST in $HOSTS; do
  echo "Injecting key into $HOST"
  ssh -o StrictHostKeyChecking=no -i $PEM_FILE $USER@$HOST "
    mkdir -p ~/.ssh && \
    echo \"$PUB_KEY\" >> ~/.ssh/authorized_keys && \
    chmod 700 ~/.ssh && \
    chmod 600 ~/.ssh/authorized_keys
  "
done
```

give permission and run file

```
chmod +x copy-public-key.sh

./copy-public-key.sh
```

# Step 8. Final Deployment Stage

## a) Give Jenkins Permission to ansible environment

```
sudo chown -R jenkins:jenkins /home/ubuntu/ansible-env

sudo chmod -R 755 /home/ubuntu/ansible-env

sudo chmod 755 /home/ubuntu
```

### verify

```
sudo -u jenkins ls -l /home/ubuntu/ansible-env/bin/activate

sudo -u jenkins bash -c "source/home/ubuntu/ansible-env/bin/activate && echo Activated"
```

```
root@ip-172-31-4-83:/home/ubuntu/k8s# sudo -u jenkins bash -c "source /home/ubunt
u/ansible-env/bin/activate && echo Activated"
Activated
```

```
sudo usermod -aG docker jenkins

sudo systemctl restart Jenkins
```

[redacted]

```
# Inside ansible-env

source /home/ubuntu/ansible-env/bin/activate

pip install --upgrade pip

pip install ansible boto3 botocore kubernetes

ansible-galaxy collection install
kubernetes.core

deactivate
```

[redacted]

[redacted]

[redacted]k8s folder and makes
files in it

Make files on forked repository

```
mkdir k8s
```

```
vi k8s/deployment.yaml

apiVersion: apps/v1

kind: Deployment

metadata:

  name: apache-deployment

spec:

  replicas: 2

  selector:

    matchLabels:

      app: apache

  template:

    metadata:

      labels:

        app: apache

    spec:

      containers:

        - name: apache
```

```
          image: <your_docker_id>/apachewebsite:latest  # Your Docker Hub image
          ports:
            - containerPort: 80
```

```
vi k8s/service.yaml
```

```
apiVersion: v1
kind: Service
metadata:
  name: apache-service
spec:
  type: LoadBalancer
  selector:
    app: apache
  ports:
    - protocol: TCP
```

```
      port: 80
      targetPort: 80

[redacted]

- name: Deploy Apache website to EKS
  hosts: localhost
  connection: local
  tasks:

    - name: Ensure AWS CLI is configured for EKS
      command: aws eks update-kubeconfig --name <your_cluster_name> --region <your_region_name>

    - name: Deploy Apache Deployment
      kubernetes.core.k8s:
        state: present
        namespace: default
```

```
        src: k8s/deployment.yaml

    - name: Deploy Apache Service
      kubernetes.core.k8s:
        state: present
        namespace: default
        src: k8s/service.yaml

    - name: Verify deployment pods
      command: kubectl get pods -o wide
      register: pod_status

    - debug:
        var: pod_status.stdout_lines
[redacted]
[redacted]
pipeline {
    agent any
```

```
    environment {
        DOCKER_IMAGE = "<docker_id>/apachewebsite:latest"
    }

    stages {
        stage('Checkout SCM') {
            steps {
                checkout scm
            }
        }

        stage('Docker Build & Push') {
            steps {
                script {

withDockerRegistry([credentialsId:
```

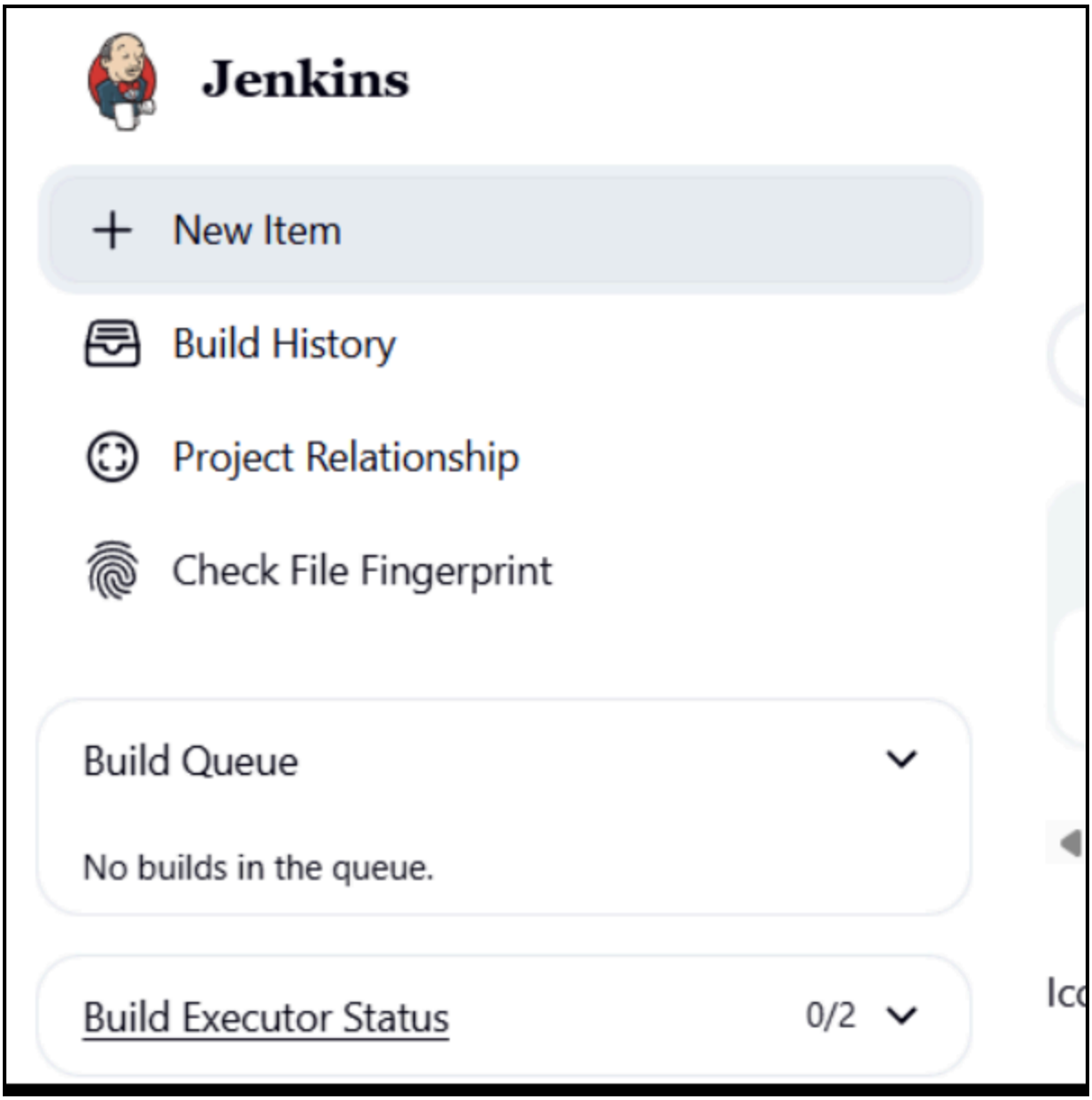
```
'dockerhub-cred-id', url:
'https://index.docker.io/v1/']) {
                                sh '''
                                    echo "Building
Docker image..."
                                    docker build -t
$DOCKER_IMAGE .

                                    echo "Pushing Docker
image to DockerHub..."
                                    docker push
$DOCKER_IMAGE
                                '''
                            }
                        }
                    }
                }

                stage('Prepare Kubernetes Manifests') {
```

```
            steps {
                sh '''
                    echo "Copying Kubernetes manifests into Jenkins workspace..."
                    mkdir -p k8s
                    cp /home/ubuntu/k8s/service.yaml k8s/service.yaml
                    cp /home/ubuntu/k8s/deployment.yaml k8s/deployment.yaml
                '''
            }
        }

        stage('Deploy to EKS via Ansible') {
            steps {
                withCredentials([
                    string(credentialsId: 'aws-access-key', variable: 'AWS_ACCESS_KEY_ID'),
```

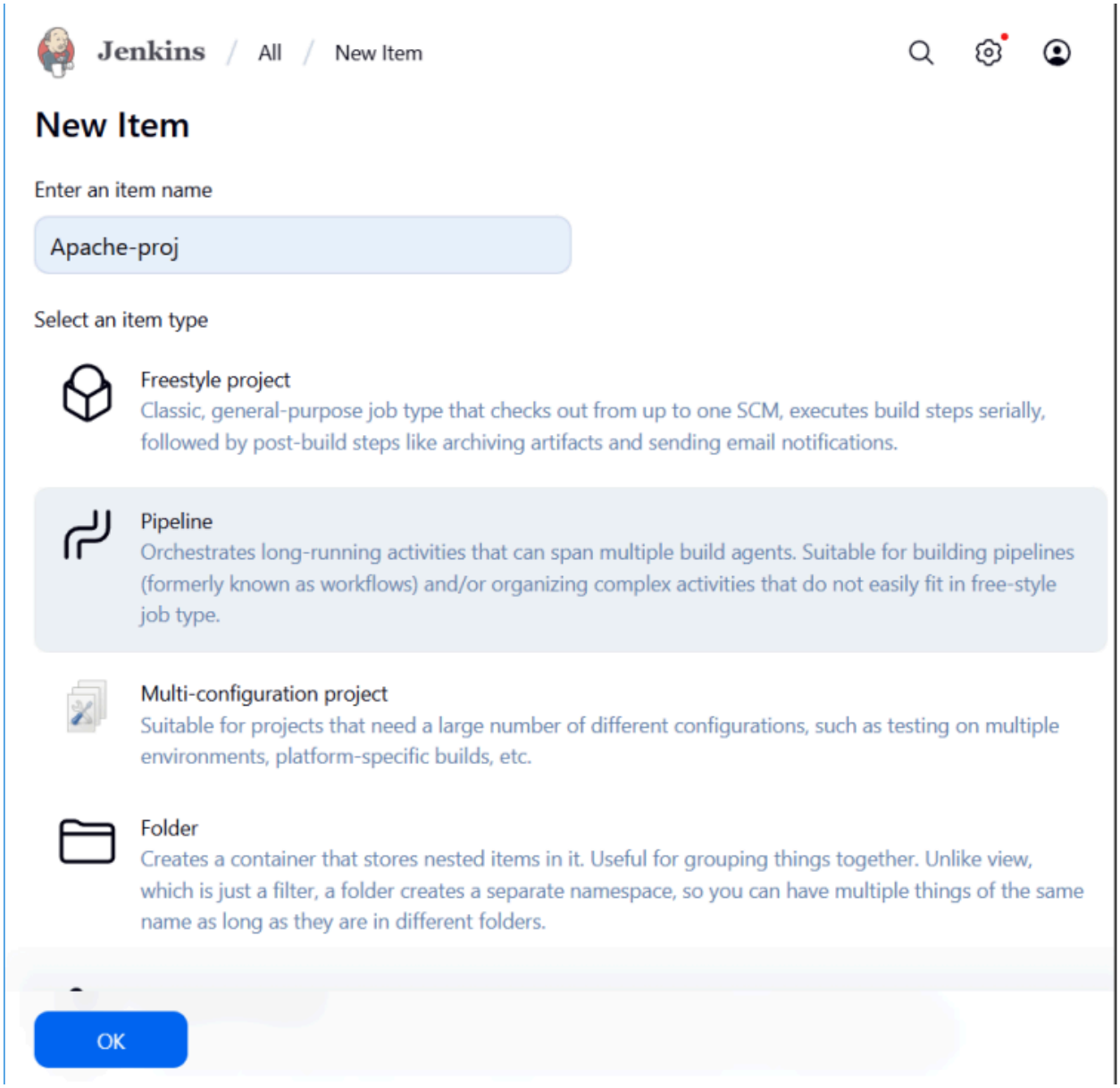
```
                    string(credentialsId:
'aws-secret-key', variable:
'AWS_SECRET_ACCESS_KEY')

                ]) {

                    sh '''

                        echo "Exporting AWS
credentials..."

                        export
AWS_ACCESS_KEY_ID=$AWS_ACCESS_KEY_ID

                        export
AWS_SECRET_ACCESS_KEY=$AWS_SECRET_ACCESS_KEY

                        export
AWS_DEFAULT_REGION=ap-south-1


                        echo "Activating Ansible
virtual environment..."

                        .
/home/ubuntu/ansible-env/bin/activate


                        echo "Running Ansible
playbook for Kubernetes deployment..."
```

```
                    ansible-playbook
deploy-k8s.yaml
                '''
            }
        }
    }

    stage('Verify Deployment') {
        steps {
            echo "Stage skipped if
deployment failed"
        }
    }
}

post {
    success {
        echo "✅ Deployment succeeded!"
```

```
        }
        failure {
            echo "❌ Deployment failed!"
        }
    }
}
```




Enter name -> Select Pipeline option -> click on OK

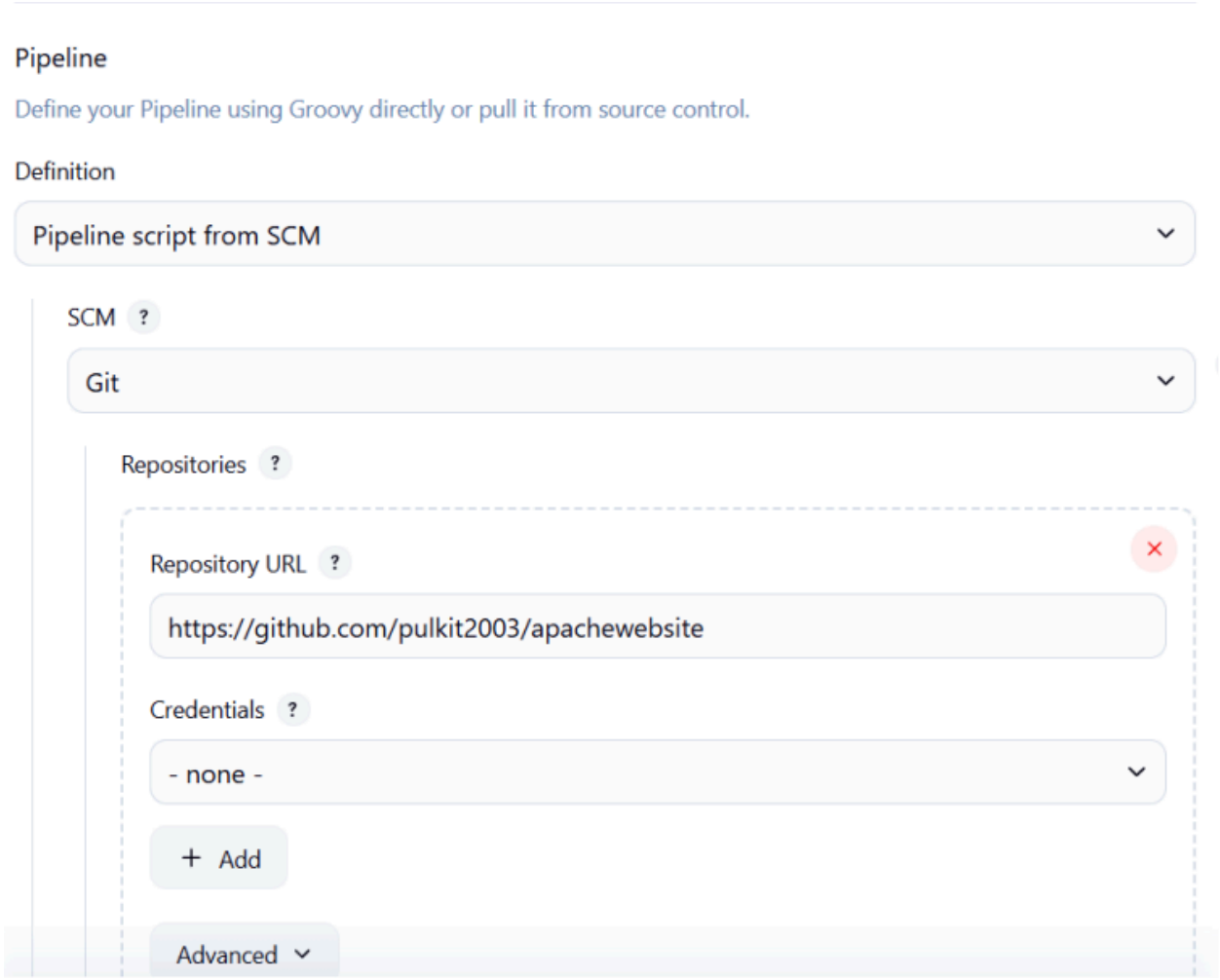



Scroll down to Pipeline section

Select Defination -> Pipeline script from SCM

Select SCM as Git

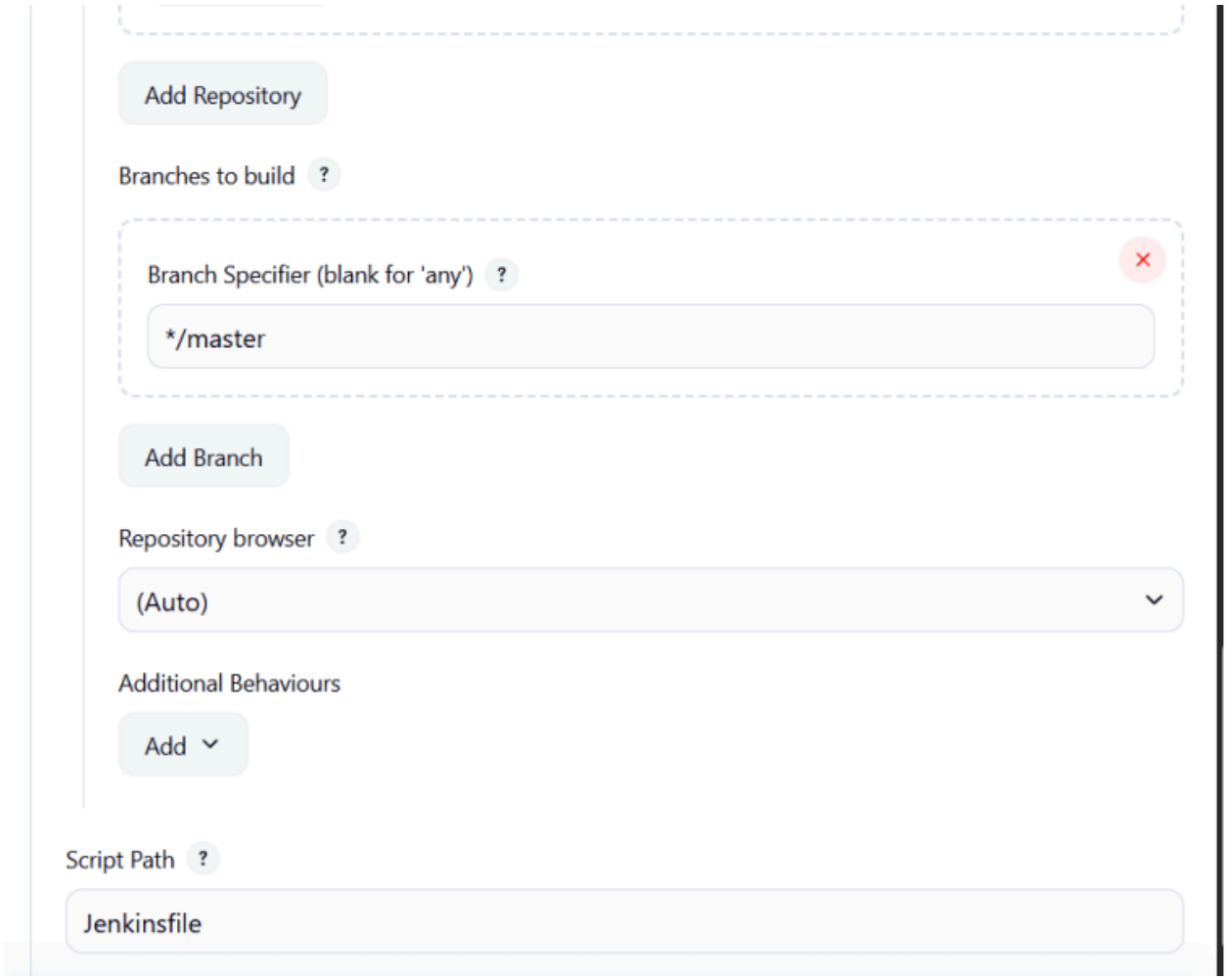
Under Repositories Add <your repository url>



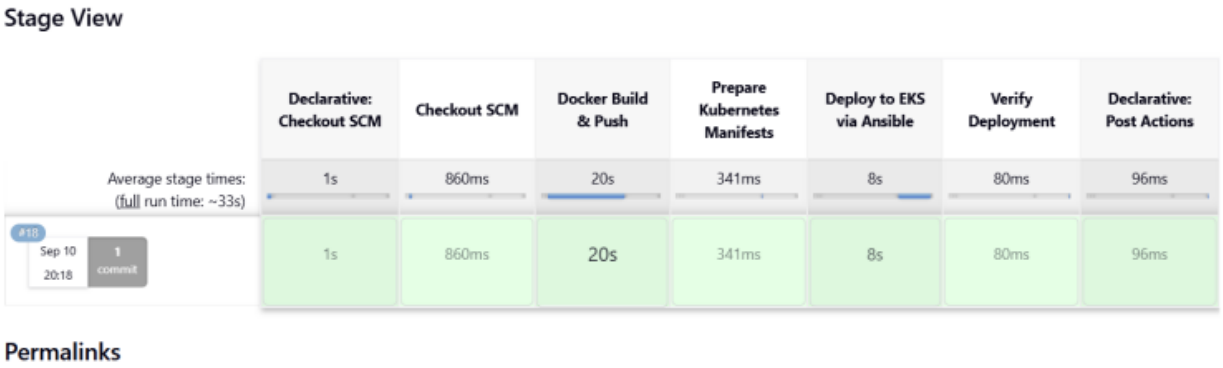

Select Master as Branch

Script path as Jenkinfile

Then click Apply and Save

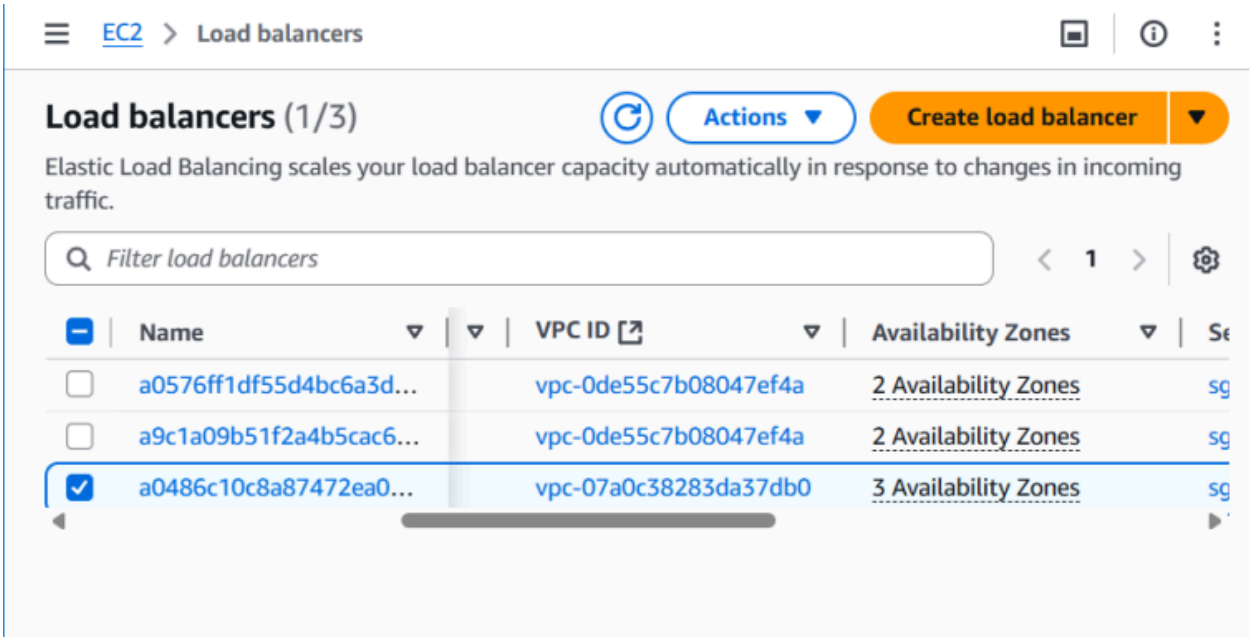


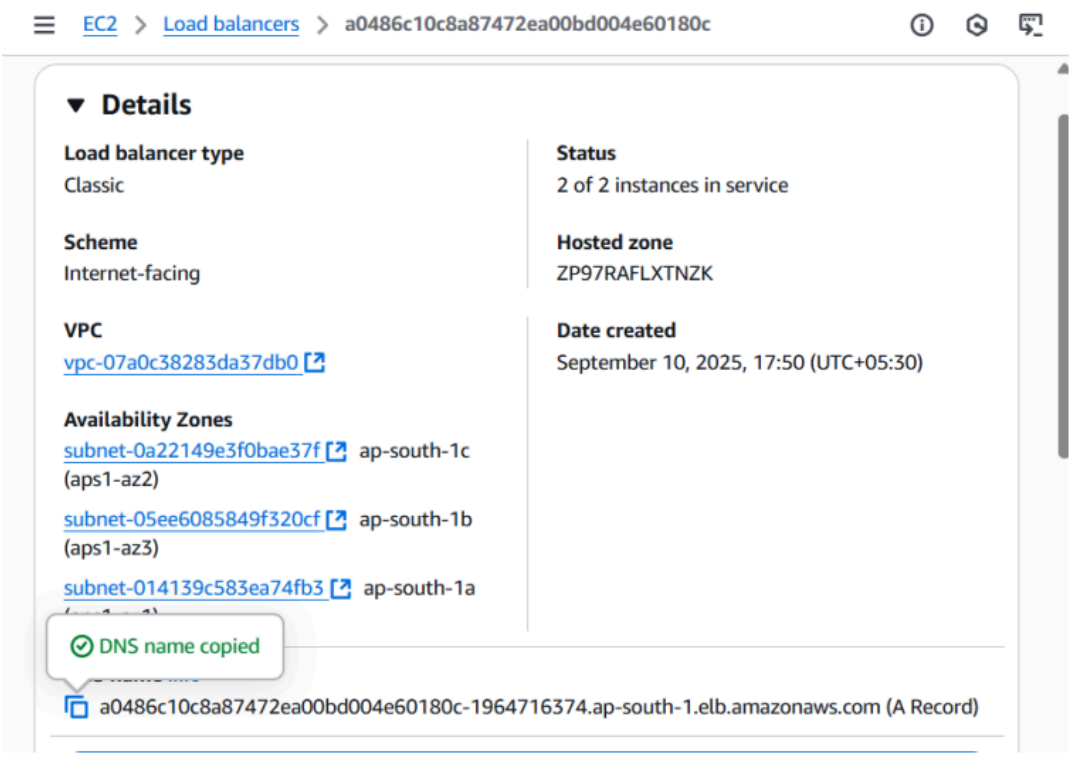

b)Run pipeline




# Step 10. Confirmation

a) Open Load Balancer and select your Load Balancer that has been created Copy the DNS link and open on Browser



# The site is running